

للتعلم العميق

باستخدام بايثون

اعداد: آمان خروال

ترجمة : د. علاء طعيمة

بسمه تعالى

20
مشروعاً للتعلم العميق باستخدام بايثون

تأليف:
أمان خروال

ترجمة:
د. علاء طعيمة

مقدمة المؤلف

في هذه الكتاب، سوف يأخذك المؤلف خلال 20 مشروعًا من مشروعات التعلم العميق باستخدام لغة برمجة بايثون التي تم حلها وشرحها مجانًا.

التعلم العميق هو مجموعة فرعية من الذكاء الاصطناعي، وهو مجال يعتمد على التعلم والتحسين من تلقاء نفسه من خلال فحص خوارزميات الكمبيوتر. بينما يستخدم التعلم الآلي مفاهيم أبسط، تعمل هذه النماذج مع الشبكات العصبية الاصطناعية، المصممة لتقليد طريقة تفكير البشر وتعلمهم.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاً مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخل في مجالات التعلم الآلي والتعلم العميق وعلم البيانات ومساعدة القارئ العربي على تعلم هذا المجالات. اسأل الله التوفيق في هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصين في مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

20 مشروعاً في التعلم العميق باستخدام بايثون

في هذه الكتاب، سوف آخذك خلال 20 مشروعًا من مشروعات التعلم العميق باستخدام لغة برمجة بايثون التي تم حلها وشرحها مجانًا.

التعلم العميق هو مجموعة فرعية من الذكاء الاصطناعي، وهو مجال يعتمد على التعلم والتحسين من تلقاء نفسه من خلال فحص خوارزميات الكمبيوتر. بينما يستخدم التعلم الآلي مفاهيم أبسط، تعمل هذه النماذج مع الشبكات العصبية الاصطناعية، المصممة لتقليد طريقة تفكير البشر وتعلمهم.

1) كشف الالتهاب الرئوي مع بايثون Pneumonia Detection with Python

في هذه المقالة، سوف أقدم لكم مشروع التعلم الآلي عن اكتشاف الالتهاب الرئوي باستخدام لغة برمجة بايثون. الالتهاب الرئوي Pneumonia هو حالة التهابية تصيب الرئة تؤثر بشكل رئيسي على الأكياس الهوائية الصغيرة المسماة الحويصلات الهوائية alveoli.

مقدمة في اكتشاف الالتهاب الرئوي

الالتهاب الرئوي هو مرض تنفسي معدي وقاتل تسببه بكتيريا أو فطريات أو فيروس يصيب الأكياس الهوائيةفي الرئة البشرية بحمل مليء بالسوائل أو الصديد.

الأشعة السينية للصدر Chest x-rays هي الطريقة الشائعة المستخدمة لتشخيص الالتهاب الرئوي ويستغرق الأمر خبيرًا طبيًا لتقييم نتيجة الأشعة السينية. تؤدي الطريقة المزعجة للكشف عن الالتهاب الرئوي إلى خسائرفي الأرواح بسبب التشخيص والعلاج غير الصحيحين.

مع قوة الحوسبة الناشئة، أصبح من الممكن الآن تطوير نظام الكشف التلقائي عن الالتهاب الرئوي وعلاج الأمراض، خاصة إذا كان المريضفي منطقة نائية وكانت الخدمات الطبية محدودة.

مشروع التعلم الآلي لاكتشاف الالتهاب الرئوي باستخدام بايثون

في هذا القسم، سوف اخوضفي مشروع تعلم الآلة حول اكتشاف الالتهاب الرئوي باستخدام لغة برمجة بايثون. سأستخدم مكتبة Fastaiفي بايثون لمهمة اكتشاف الالتهاب الرئوي.

لنبدأ الآن بهذه المهمة عن طريق استيراد مكتبات بايثون الضرورية:

تحميل مجموعة البيانات

```
from fastai.vision import *
```

```
from fastai.metrics import
error_rate
import os
import pandas as pd
import numpy as np
```

نحتاج الآن إلى إعداد مسار مجموعة بيانات التدريب حيث تتضمن مجموعة البيانات الصور فقط:

```
x  = 'Path'
path = Path(x)
path.ls()
```

لنقم الآن بتحميل بيانات التدريب أو نموذج التعلم الآلي لمهمة اكتشاف الالتهاب الرئوي باستخدام بايثون:

```
np.random.seed(40)
data = ImageDataBunch.from_folder(path, train = '.', valid_pct=0.2,
                ds_tfms=get_transforms(), size=224,
                num_workers=4).normalize(imagenet_stats)
```

استكشاف البيانات

يتم تخزين مجموعة البيانات التي أستخدمها هنا كملفات jpg.في مجلدين مختلفين، يحتوي كل مجلد على اسم طراز الصورفي المجلد.

نحتاج إلى استخدام دالة ImageDataBunch.from_folder() لتحميل الصور وتعيين علامات للصور بناءً على اسم المجلد الذي تمت قراءتها منه:

```
data.show_batch(rows=3, figsize=(7,6),recompute_scale_factor=True)
```

PNEUMONIA NORMAL PNEUMONIA
PNEUMONIA NORMAL PNEUMONIA
PNEUMONIA NORMAL PNEUMONIA

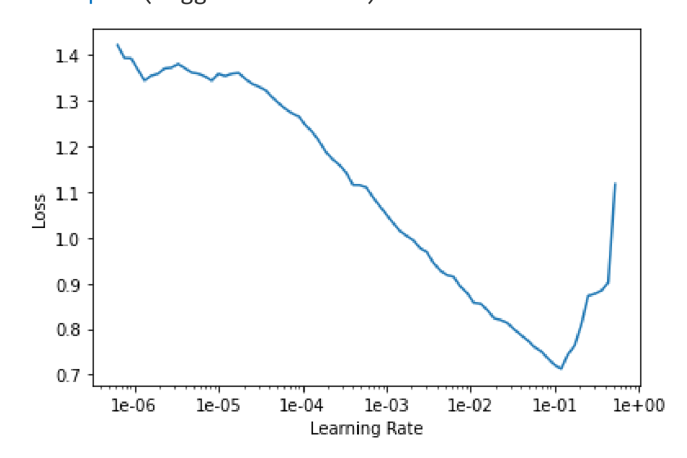
استخدام التعلم الآلي للكشف عن الالتهاب الرئوي باستخدام لغة بايثون

الآن، سأستخدم نموذجًا تم تدريبه مسبقًا يُعرف باسم ResNet50، وهو نوع من الشبكات العصبية التلافيفية CNN في التعلم الآلي. الآن دعونا نرى كيفية استخدام هذا النموذج:

```
learn = cnn_learner(data, models.resnet50, metrics=[accuracy],
model_dir = Path('Path'),path = Path("."))
```

دعنا الآن نلقي نظرة على معدل التعلم learning rate للنموذج:

```
learn.lr_find()
learn.recorder.plot(suggestions=True)
```



تدريب واختبار النموذج

في القسم أعلاه، قمنا بتحميل النموذج. الآن سأقوم بتدريب النموذج على مجموعة البيانات الخاصة بنا:

```
lr1 = 1e-3
lr2 = 1e-1
learn.fit_one_cycle(4,slice(lr1,lr2))

# lr1 = 1e-3
lr = 1e-1
learn.fit_one_cycle(20,slice(lr))

learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
learn.fit_one_cycle(10,slice(1e-4,1e-3))
```

```
learn.recorder.plot_losses()
```

الآن دعنا نختبر النموذج:

```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
img = open_image('IM-0001-0001.jpeg')
print(learn.predict(img)[0])
NORMAL
```

هذه هي الطريقة التي يمكننا بها استخدام التعلم الآلي للكشف عن الالتهاب الرئوي. آمل أن تكون قد أحببت هذه المقالة حول مشروع التعلم الالي حول اكتشاف الالتهاب الرئوي باستخدام بايثون.

2) اكتشاف قناع الوجه مع التعلم الآلي Face Mask Detection with Machine Learning

أصبح اكتشاف الوجه مشكلة مثيرة للاهتمام للغايةفي معالجة الصور والرؤية الحاسوبية.في هذه المقالة، سأقدم لك مشروع الرؤية الحاسوبية حول اكتشاف قناع الوجه باستخدام التعلم الآلي باستخدام بايثون.

مقدمة في اكتشاف قناع الوجه

يحتوي اكتشاف قناع الوجه Face mask detection على مجموعة من التطبيقات بدءًا من التقاط حركة الوجه وحتى التعرف على الوجه والذي يتطلبفي البداية اكتشاف الوجه بدقة جيدة جدًا. يعد اكتشاف الوجه Face detection أكثر أهمية اليوم لأنه لا يستخدم فقطفي الصور، ولكن أيضًافي تطبيقات الفيديو مثل المراقبةفي الوقت الفعلي واكتشاف الوجهفي مقاطع الفيديو.

أصبح تصنيف الصور عالي الدقة ممكنًا الآن مع التقدمفي الشبكات العصبية التلافيفية CNN. غالبًا ما تكون المعلومات المتعلقة بمستوى البكسل مطلوبة بعد اكتشاف الوجه، والتي لا توفرها معظم طرق اكتشاف الوجه.

كان الحصول على تفاصيل على مستوى البكسل جزءًا صعبًا من التجزئة الدلالية semantic segmentation. التجزئة الدلالية هي عملية تعيين تسمية لكل بكسلفي الصورة.

عملية اكتشاف قناع الوجه باستخدام التعلم الآلي

الخطوة 1: استخراج بيانات الوجه للتدريب.

الخطوة 2: تدريب المصنف على تصنيف الوجوهفي قناع أو تسميات بدون قناع.

الخطوة 3: اكتشاف الوجوه أثناء اختبار البيانات باستخدام كاشف الوجه SSD.

الخطوة 4: باستخدام المصنف المدرب، صنف الوجوه المكتشفة.

في الخطوة الثالثة من العملية المذكورة أعلاه، عليك التفكير ما هو كاشف الوجه SSD؟ حسنًا، SSD عبارة عن كاشف متعدد الصناديق ذو اللقطة الواحدة Single Shot Multibox Detector. هذه تقنية تستخدم لاكتشاف الأشياءفي الصور باستخدام شبكة عصبية واحدة عميقة.

يتم استخدامه للكشف عن الأشياءفي الصورة. باستخدام بُنية أساسية لبنية VGG-16، يمكن أن يتفوق SSD على أجهزة الكشف عن الكائنات الأخرى مثل YOLO وFaster R-CNN من حيث السرعة والدقة.

اكتشاف قناع الوجه مع التعلم الآلي

الآن، دعنا نبدأ بمهمة اكتشاف قناع الوجه باستخدام التعلم الآلي باستخدام لغة برمجة بايثون.
سأبدأ هذه المهمة عن طريق استيراد مكتبات بايثون الضرورية التي نحتاجها لهذه المهمة:

تحميل مجموعة البيانات

```
import pandas as pd
import numpy as np
import cv2
import json
import os
import matplotlib.pyplot as plt
import random
import seaborn as sns
from keras.models import Sequential
from keras import optimizers
from keras import backend as K
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
directory = "../input/face-mask-detection-dataset/Medical mask/Medical
mask/Medical Mask/annotations"
image_directory = "../input/face-mask-detection-dataset/Medical
mask/Medical mask/Medical Mask/images"
df = pd.read_csv("../input/face-mask-detection-dataset/train.csv")
df_test = pd.read_csv("../input/face-mask-detection-
dataset/submission.csv")
```

إنشاء دوال المساعدة

سأبدأ هذه المهمة بإنشاء دالتين مساعدتين:

```
cvNet = cv2.dnn.readNetFromCaffe('weights.caffemodel')
def getJSON(filePathandName):
    with open(filePathandName,'r') as f:
        return json.load(f)
def adjust_gamma(image, gamma=1.0):
    invGamma = 1.0 / gamma
    table = np.array([((i / 255.0) ** invGamma) * 255 for i in
np.arange(0, 256)])
```

```
    return cv2.LUT(image.astype(np.uint8), table.astype(np.uint8))
```

1. تسترد الدالة getJSON ملف json الذي يحتوي على بيانات الصندوق المحيط في مجموعة بيانات التدريب.
2. تعد دالة Adjust_gamma عملية غير خطية تُستخدم لترميز وفك ترميز قيم luminance أو tristimulus في أنظمة الفيديو أو الصور الثابتة. ببساطة، يتم استخدامه لغرس القليل من الضوء في الصورة. إذا كانت gamma <1، فستتحول الصورة إلى الطرف الأغمق من الطيف وعندما تكون 1 <gamma، سيكون هناك المزيد من الضوء في الصورة.

معالجة البيانات

الخطوة التالية الآن هي استكشاف بيانات JSON المقدمة للتدريب:

```
Jsonfiles=[]

for i in os.listdir(directory):

jsonfiles.append(getJSON(os.path.join(directory,i)))

jsonfiles[0]
```


```
{'FileName': '2349.png',
 'NumOfAnno': 4,
 'Annotations': [{'isProtected': False,
  'ID': 193452793312540288,
  'BoundingBox': [29, 69, 285, 343],
  'classname': 'face_other_covering',
  'Confidence': 1,
  'Attributes': {}},
 {'isProtected': False,
  'ID': 545570408121800384,
  'BoundingBox': [303, 99, 497, 341],
  'classname': 'face_other_covering',
  'Confidence': 1,
  'Attributes': {}},
 {'isProtected': False,
  'ID': 339053397051370048,
  'BoundingBox': [8, 71, 287, 373],
  'classname': 'hijab_niqab',
  'Confidence': 1,
  'Attributes': {}},
 {'isProtected': False,
  'ID': 100482004994698944,
  'BoundingBox': [296, 99, 525, 371],
  'classname': 'hijab_niqab',
  'Confidence': 1,
  'Attributes': {}}]}
```

- يحتوي حقل التعليقات التوضيحية Annotations field على بيانات جميع الوجوه الموجودة في صورة معينة.
- توجد أسماء فئات مختلفة، لكن أسماء الفئات الحقيقية هي face_with_mask و face_no_mask.

```
df = pd.read_csv("train.csv")
df.head()
```

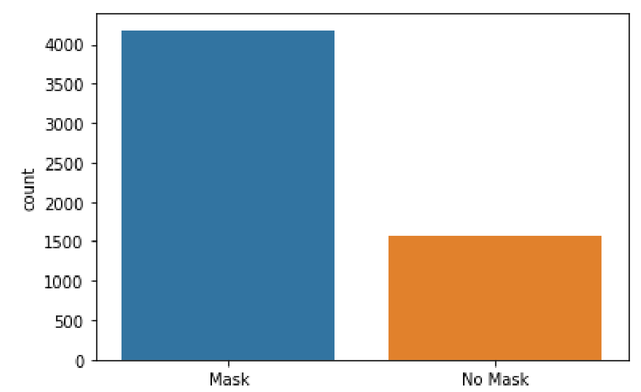
	name	x1	x2	y1	y2	classname
0	2756.png	69	126	294	392	face_with_mask
1	2756.png	505	10	723	283	face_with_mask
2	2756.png	75	252	264	390	mask_colorful
3	2756.png	521	136	711	277	mask_colorful
4	6098.jpg	360	85	728	653	face_no_mask

باستخدام القناع والتسميات non_mask، يتم استخراج بيانات الصندوق المحيط لملفات json. يتم استخراج وجوه صورة معينة وتخزينهافي قائمة البيانات مع علامتها لعملية التعلم.

```
data = []
img_size = 124
mask = ['face_with_mask']
non_mask = ["face_no_mask"]
labels={'mask':0,'without mask':1}
for i in df["name"].unique():
    f = i+".json"
    for j in getJSON(os.path.join(directory,f)).get("Annotations"):
        if j["classname"] in mask:
            x,y,w,h = j["BoundingBox"]
            img = cv2.imread(os.path.join(image_directory,i),1)
            img = img[y:h,x:w]
            img = cv2.resize(img,(img_size,img_size))
            data.append([img,labels["mask"]])
        if j["classname"] in non_mask:
            x,y,w,h = j["BoundingBox"]
            img = cv2.imread(os.path.join(image_directory,i),1)
```

```
            img = img[y:h,x:w]
            img = cv2.resize(img,(img_size,img_size))
            data.append([img,labels["without mask"]])
random.shuffle(data)

p = []
for face in data:
    if(face[1] == 0):
        p.append("Mask")
    else:
        p.append("No Mask")
sns.countplot(p)
```



يخبرنا الرسم أعلاه أن عدد صور القناع> عدد الصور بدون قناع، لذا فهذه مجموعة بيانات غير متوازنة. ولكن نظرًا لأننا نستخدم نموذج SSD مُدرَّبًا مسبقًا، ومُدرَّبًا على اكتشاف الوجوه غير المقنعة، فإن هذا الاختلال في التوازن لن يكون مهمًا كثيرًا.

لكن دعونا نعيد تشكيل البيانات قبل تدريب الشبكة العصبية:

```
X = []
Y = []
for features,label in data:
    X.append(features)
    Y.append(label)

X = np.array(X)/255.0
X = X.reshape(-1,124,124,3)
```

```
Y = np.array(Y)
```

شبكة تدريب عصبية لاكتشاف قناع الوجه

الآن الخطوة التالية هي تدريب شبكة عصبية على مهمة اكتشاف قناع الوجه باستخدام التعلم الآلي:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding = "same", activation='relu',
input_shape=(124,124,3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam'
,metrics=['accuracy'])
xtrain,xval,ytrain,yval=train_test_split(X,
Y,train_size=0.8,random_state=0)
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=15,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True,
        vertical_flip=False)
datagen.fit(xtrain)

history = model.fit_generator(datagen.flow(xtrain, ytrain,
batch_size=32),
                    steps_per_epoch=xtrain.shape[0]//32,
```

```
            epochs=50,
            verbose=1,
            validation_data=(xval, yval))
```

اختبار النموذج

تحتوي مجموعة بيانات الاختبار على 1698 صورة ولتقييم النموذج، لذا التقطت عددًا قليلاً من الصور من مجموعة البيانات هذه نظرًا لعدم وجود علامات وجه face tagsفي مجموعة البيانات:

```
test_images = ['1114.png','1504.jpg',
'0072.jpg','0012.jpg','0353.jpg','1374.jpg']
gamma = 2.0
fig = plt.figure(figsize = (14,14))
rows = 3
cols = 2
axes = []
assign = {'0':'Mask','1':"No Mask"}
for j,im in enumerate(test_images):
    image =  cv2.imread(os.path.join(image_directory,im),1)
    image =  adjust_gamma(image, gamma=gamma)
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(image, (300,300)), 1.0,
(300, 300), (104.0, 177.0, 123.0))
    cvNet.setInput(blob)
    detections = cvNet.forward()
    for i in range(0, detections.shape[2]):
        try:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            frame = image[startY:endY, startX:endX]
            confidence = detections[0, 0, i, 2]
            if confidence > 0.2:
                im = cv2.resize(frame,(img_size,img_size))
                im = np.array(im)/255.0
                im = im.reshape(1,124,124,3)
                result = model.predict(im)
                if result>0.5:
                    label_Y = 1
                else:
```

```
                        label_Y = 0
                    cv2.rectangle(image, (startX, startY), (endX, endY), (0,
0, 255), 2)
                    cv2.putText(image,assign[str(label_Y)] , (startX,
startY-10), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (36,255,12), 2)

        except:pass
    axes.append(fig.add_subplot(rows, cols, j+1))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.show()
```

من خلال تحليل الناتج أعلاه، يمكننا أن نلاحظ أن النظام بأكمله يعمل بشكل جيد مع الوجوه التي لها سيطرة مكانية spatial dominance. لكنها تفشل في حالة الصور التي تكون فيها الوجوه صغيرة وتشغل مساحة أقل في الصورة الإجمالية.

للحصول على أفضل النتائج، يمكن استخدام تقنيات مختلفة للمعالجة المسبقة للصور، أو يمكن إبقاء حد الثقة منخفضًا، أو يمكن للمرء تجربة أحجام blob مختلفة.

3) نموذج التنبؤ بالزلازل مع التعلم الآلي Earthquake Prediction Model with Machine Learning

في هذه المقالة، سأطلعك على كيفية إنشاء نموذج لمهمة توقع الزلازل باستخدام التعلم الآلي ولغة برمجة بايثون. يعد التنبؤ بالزلازل أحد أكبر المشكلات التي لم يتم حلهافي علوم الأرض.

مع زيادة استخدام التكنولوجيا، زادت العديد من محطات المراقبة الزلزالية، لذلك يمكننا استخدام التعلم الآلي والأساليب الأخرى التي تعتمد على البيانات للتنبؤ بالزلازل.

نموذج التنبؤ بالزلازل مع التعلم الآلي

من المعروف أنه إذا حدثت كارثةفي منطقة ما، فمن المحتمل أن تحدث مرة أخرى. بعض المناطق بها زلازل متكررة، ولكن هذا ليس سوى كمية مقارنة بالمقارنة مع المناطق الأخرى.

لذلك، فإن التنبؤ بالزلزال مع التاريخ والوقت وخط العرض وخط الطول من البيانات السابقة ليس اتجاهًا يتبع مثل الأشياء الأخرى، إنه يحدث بشكل طبيعي.

سأبدأ هذه المهمة لإنشاء نموذج للتنبؤ بالزلازل عن طريق استيراد مكتبات بايثون الضرورية:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

فلنقم الآن بتحميل مجموعة البيانات وقراءتها. يمكن تنزيل مجموعة البيانات التي أستخدمها هنا بسهولة من هنا:

```
data = pd.read_csv("database.csv")
data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status'],
      dtype='object')
```

الآن دعنا نرى الخصائص الرئيسية لبيانات الزلازل وننشئ كائنًا من هذه الخصائص، أي التاريخ والوقتdate time وخط العرض latitude وخط الطول longitude والعمق depth والحجم magnitude:

```
data = data [['Date'،'Time'،'Latitude'،'Longitude'،'Depth'،
'Magnitude']]
```

```
data.head()
```

	date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

نظرًا لأن البيانات عشوائية، فنحن بحاجة إلى قياسها بناءً على مدخلات النموذج.في هذا، نقوم بتحويل التاريخ والوقت المحددين إلى وقت Unix وهو بالثواني ورقم. يمكن استخدام هذا بسهولة كمدخل للشبكة التي أنشأناها:

```
import datetime
import time

timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # print('ValueError')
        timestamp.append('ValueError')
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

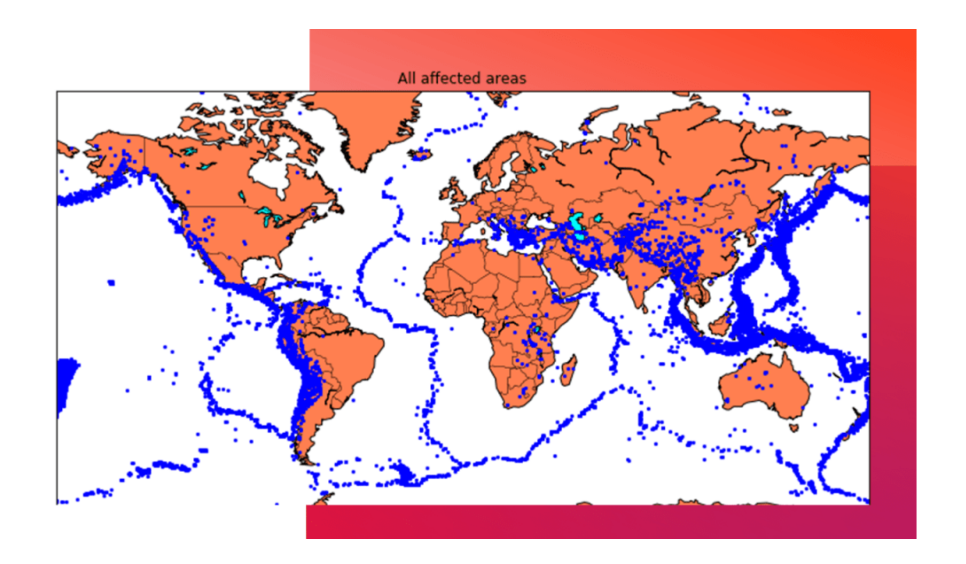
العرض المرئي للبيانات

الآن، قبل إنشاء نموذج التنبؤ بالزلازل، دعنا نرسم البيانات الموجودة على خريطة العالم التي تعرض تمثيلًا واضحًا لمكان تواتر الزلازل:

```
from mpl_toolkits.basemap import Basemap
m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-
180,urcrnrlon=180,lat_ts=20,resolution='c')

longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
            #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)

fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```



تقسيم مجموعة البيانات

الآن، لإنشاء نموذج التنبؤ بالزلازل، نحتاج إلى تقسيم البيانات إلى Xs و ys والتي سيتم إدخالها على التوالي في النموذج كمدخلات لتلقي الإخراج من النموذج.

المدخلات هنا هي TImestamp وLatitude وLongitude والمخرجات هي Magnitude وDepth. سأقوم بتقسيم x وy إلى تدريب train واختبار test مع التحقق من الصحة validation. تحتوي مجموعة التدريب على 80٪ ومجموعة الاختبار تحتوي على 20٪:

```
X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

```
(18727, 3) (4682, 3) (18727, 2) (4682, 3)
```

الشبكة العصبية للتنبؤ بالزلازل

الآن سوف أقوم بإنشاء شبكة عصبية لتناسب البيانات من مجموعة التدريب. ستتألف شبكتنا العصبية من ثلاث طبقات كثيفة dense layers تحتوي كل منها على 16 و16 و2 عقدة وتعيد قراءتها. سيتم استخدام Relu وsoftmax كدوال تنشيط:

```
from keras.models import Sequential
from keras.layers import Dense

def create_model(neurons, activation, optimizer, loss):
    model = Sequential()
    model.add(Dense(neurons, activation=activation, input_shape=(3,)))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(2, activation='softmax'))
    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

سأقوم الآن بتحديد المعلمات الفائقة hyperparameters بخيارين أو أكثر للعثور على أفضل ملاءمة best fit:

```
from keras.wrappers.scikit_learn import KerasClassifier
model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
neurons = [16]
```

```
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear',
'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam',
'Adamax', 'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']

param_grid = dict(neurons=neurons, batch_size=batch_size,
epochs=epochs, activation=activation, optimizer=optimizer, loss=loss)
```

نحتاج الآن إلى العثور على أفضل نموذج ملائم للنموذج أعلاه والحصول على متوسط درجة الاختبار والانحراف المعياري لأفضل نموذج مناسب:

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-
1)
grid_result = grid.fit(X_train, y_train)


print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.957655 using {'activation': 'relu', 'batch_size': 10,
'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer':
'SGD'} 0.333316 (0.471398) with: {'activation': 'sigmoid',
'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons':
16, 'optimizer': 'SGD'} 0.000000 (0.000000) with: {'activation':
'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge',
'neurons': 16, 'optimizer': 'Adadelta'} 0.957655 (0.029957) with:
{'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'} 0.645111
(0.456960) with: {'activation': 'relu', 'batch_size': 10, 'epochs':
10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
```

في الخطوة أدناه، يتم استخدام أفضل المعلمات الملائمة لنفس النموذج لحساب النتيجة باستخدام بيانات التدريب وبيانات الاختبار:

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(3,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='SGD', loss='squared_hinge',
metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=10, epochs=20, verbose=1,
validation_data=(X_test, y_test))

[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy =
{}".format(test_loss, test_acc))
```

```
Evaluation result on Test Data : Loss = 0.5038455790406056, accuracy
= 0.9241777017858995
```

لذلك يمكننا أن نرى في الناتج أعلاه أن نموذج الشبكة العصبية الخاص بنا للتنبؤ بالزلازل يعمل بشكل جيد. آمل أن تكون قد أحببت هذه المقالة حول كيفية إنشاء نموذج التنبؤ بالزلازل باستخدام التعلم الآلي ولغة برمجة بايثون.

4) اكتشاف المعالم باستخدام التعلم الآلي Landmark Detection with Machine Learning

هل سبق لك أن نظرت في صور عطلتك وتساءلت: ما اسم هذا المعبد الذي زرته في الهند؟ من أنشأ هذا النصب الذي رأيته في كاليفورنيا؟ يمكن أن يساعدنا اكتشاف المعالم Landmark Detection في اكتشاف أسماء هذه الأماكن. لكن كيف يعمل الكشف عن المعالم؟ في هذه المقالة، سأقدم لك مشروع التعلم الآلي حول اكتشاف المعالم باستخدام بايثون.

ما هو اكتشاف المعالم؟

اكتشاف المعالم هي مهمة الكشف عن التماثيل والهياكل والمعالم الأثرية التي صنعها الإنسان داخل الصورة. لدينا بالفعل تطبيق مشهور جداً لمثل هذه المهام والذي يُعرف عمومًا باسم Google Landmark Detection، والذي تستخدمه خرائط Google.

في نهاية هذه المقالة، ستتعرف على كيفية عمل اكتشاف معالم Google حيث سأأخذك عبر مشروع التعلم الآلي الذي يعتمد على دالة Google Landmark Detection. سأستخدم لغة برمجة بايثون لبناء شبكات عصبية لاكتشاف المعالم داخل الصور.

لنبدأ الآن في مهمة اكتشاف المعالم داخل الصورة. تتمثل أصعب مهمة في هذا المشروع في العثور على مجموعة بيانات تتضمن بعض الصور التي يمكننا استخدامها لتدريب شبكتنا العصبية.

نأمل، بعد الكثير من البحث، أن أكون قد صادفت مجموعة بيانات مقدمة من Google في مسابقات Kaggle. يمكنك تنزيل مجموعة البيانات التي سأستخدمها لاكتشاف المعالم باستخدام التعلم الآلي من هنا.

اكتشاف معالم كوكل مع التعلم الآلي

الآن لبدء هذه المهمة، سأستورد جميع مكتبات بايثون الضرورية التي نحتاجها لإنشاء نموذج تعلم آلي لمهمة اكتشاف المعالم:

```
import numpy as np
import pandas as pd
import keras
import cv2
from matplotlib import pyplot as plt
import os
import random
from PIL import Image
```

لذلك بعد استيراد المكتبات المذكورة أعلاه، فإن الخطوة التالية في هذه المهمة هي استيراد مجموعة البيانات التي سأستخدمها لاكتشاف المعالم بالصور:

```
samples = 20000
df = pd.read_csv("train.csv")
df = df.loc[:samples,:]
num_classes = len(df["landmark_id"].unique())
num_data = len(df)
```

الآن دعونا نلقي نظرة على حجم بيانات التدريب وعدد الفئات الفريدةفي بيانات التدريب:

```
print("Size of training data:", df.shape)
print("Number of unique classes:", num_classes)
```

```
Size of training data: (20001, 2)
Number of unique classes: 1020
```

هناك 20،001 عينة تدريب، تنتمي إلى 1020 فئةً، مما يعطينا متوسط 19.6 صورة لكل فئة، ومع ذلك، قد لا يكون هذا التوزيع هو الحال، لذلك دعونا نلقي نظرة على توزيع العينات حسب الفئة:

```
data = pd.DataFrame(df['landmark_id'].value_counts())
#index the data frame
data.reset_index(inplace=True)
data.columns=['landmark_id','count']

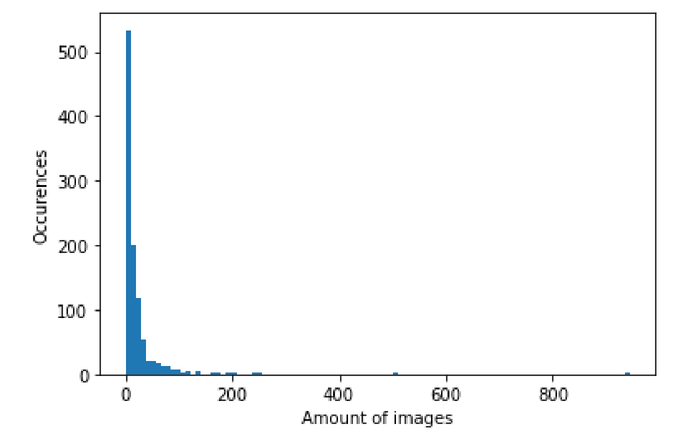
print(data.head(10))
print(data.tail(10))
```

```
landmark_id  count
0         1924    944
1           27    504
2          454    254
3         1346    244
4         1127    201
5          870    193
6         2185    177
7         1101    162
8          389    140
9          219    139
      landmark_id  count
1010          499      2
1011         1942      2
1012          875      2
1013         2297      2
1014          611      2
1015         1449      2
1016         1838      2
1017          604      2
1018          374      2
1019          991      2
```

كما نرى، تتراوح المعالم العشرة الأكثر شيوعًا من 139 نقطة بيانات إلى 944 نقطة بيانات بينما تحتوي العشر الأخيرة جميعها على نقطتي بيانات.

```
print(data['count'].describe())#statistical data for the distribution
plt.hist(data['count'],100,range = (0,944),label = 'test')#Histogram of the distribution
plt.xlabel("Amount of images")
plt.ylabel("Occurences")
```

```
count    1020.000000
mean       19.608824
std        41.653684
min         2.000000
25%         5.000000
50%         9.000000
75%        21.000000
max       944.000000
Name: count, dtype: float64
Text(0, 0.5, 'Occurences')
```



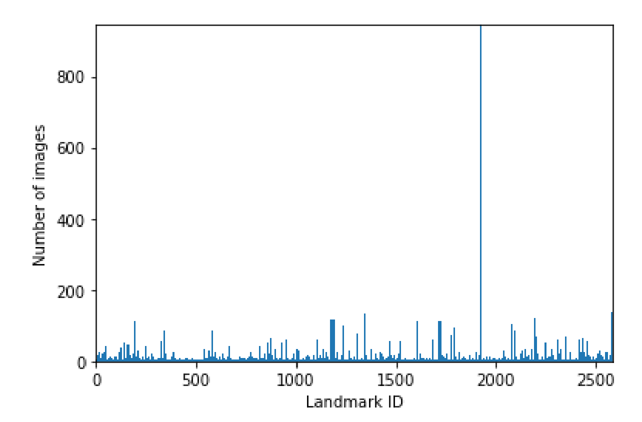
كما نرى في الرسم البياني أعلاه، فإن الغالبية العظمى من الفئات غير مرتبطة بالعديد من الصور.

```
print("Amount of classes with five and less datapoints:", (data['count'].between(0,5)).sum())
print("Amount of classes with with between five and 10 datapoints:", (data['count'].between(5,10)).sum())

n = plt.hist(df["landmark_id"],bins=df["landmark_id"].unique())
freq_info = n[0]
```

```
plt.xlim(0,data['landmark_id'].max())
plt.ylim(0,data['count'].max())
plt.xlabel('Landmark ID')
plt.ylabel('Number of images')
```

```
Amount of classes with five and less datapoints: 322
Amount of classes with with between five and 10 datapoints: 342
Text(0, 0.5, 'Number of images')
```



يوضح الرسم البياني أعلاه أن أكثر من 50٪ من فئات 1020 تحتوي على أقل من 10 صور، وهو ما قد يكون صعبًا عند تدريب المصنف.

هناك بعض "القيم المتطرفة outliers" من حيث عدد الصور لديهم، مما يعني أننا قد نكون منحازين تجاه هؤلاء، حيث قد تكون هناك فرصة أكبر للحصول على "تخمين guess" صحيح بأكبر قدرفي هذه الفئات.

تدريب النموذج

الآن، سأقوم بتدريب نموذج التعلم الآلي على مهمة الكشف عن المعالم باستخدام لغة برمجة بايثون التي ستعمل بنفس نموذج اكتشاف معالم Google.

```
from sklearn.preprocessing import LabelEncoder
lencoder = LabelEncoder()
lencoder.fit(df["landmark_id"])

def encode_label(lbl):
    return lencoder.transform(lbl)

def decode_label(lbl):
    return lencoder.inverse_transform(lbl)

def get_image_from_number(num):
```

```
    fname, label = df.loc[num,:]
    fname = fname + ".jpg"
    f1 = fname[0]
    f2 = fname[1]
    f3 = fname[2]
    path = os.path.join(f1,f2,f3,fname)
    im = cv2.imread(os.path.join(base_path,path))
    return im, label

print("4 sample images from random classes:")
fig=plt.figure(figsize=(16, 16))
for i in range(1,5):
    a = random.choices(os.listdir(base_path), k=3)
    folder = base_path+'/'+a[0]+'/'+a[1]+'/'+a[2]
    random_img = random.choice(os.listdir(folder))
    img = np.array(Image.open(folder+'/'+random_img))
    fig.add_subplot(1, 4, i)
    plt.imshow(img)
    plt.axis('off')

plt.show()
```

```
from keras.applications import VGG19
from keras.layers import *
from keras import Sequential

### Parameters
# learning_rate    = 0.0001
# decay_speed      = 1e-6
# momentum         = 0.09

# loss_function    = "sparse_categorical_crossentropy"
source_model = VGG19(weights=None)
#new_layer = Dense(num_classes, activation=activations.softmax,
name='prediction')
```

```
drop_layer = Dropout(0.5)
drop_layer2 = Dropout(0.5)


model = Sequential()
for layer in source_model.layers[:-1]: # go through until last layer
    if layer == source_model.layers[-25]:
        model.add(BatchNormalization())
    model.add(layer)
#     if layer == source_model.layers[-3]:
#         model.add(drop_layer)
# model.add(drop_layer2)
model.add(Dense(num_classes, activation="softmax"))
model.summary()


opt1 = keras.optimizers.RMSprop(learning_rate = 0.0001, momentum =
0.09)
opt2 = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9,
beta_2=0.999, epsilon=1e-07)
model.compile(optimizer=opt1,
             loss="sparse_categorical_crossentropy",
             metrics=["accuracy"])

#sgd = SGD(lr=learning_rate, decay=decay_speed, momentum=momentum,
nesterov=True)
# rms = keras.optimizers.RMSprop(lr=learning_rate, momentum=momentum)
# model.compile(optimizer=rms,
#               loss=loss_function,
#               metrics=["accuracy"])
# print("Model compiled! \n")
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization (BatchNo (None, 224, 224, 3)       12
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense (Dense)                (None, 1020)              4178940
=================================================================
Total params: 143,749,192
Trainable params: 143,749,186
Non-trainable params: 6
```

```
### Function used for processing the data, fitted into a data generator.
def get_image_from_number(num, df):
    fname, label = df.iloc[num,:]
    fname = fname + ".jpg"
    f1 = fname[0]
    f2 = fname[1]
    f3 = fname[2]
    path = os.path.join(f1,f2,f3,fname)
    im = cv2.imread(os.path.join(base_path,path))
    return im, label

def image_reshape(im, target_size):
    return cv2.resize(im, target_size)

def get_batch(dataframe,start, batch_size):
    image_array = []
    label_array = []

    end_img = start+batch_size
    if end_img > len(dataframe):
        end_img = len(dataframe)

    for idx in range(start, end_img):
        n = idx
        im, label = get_image_from_number(n, dataframe)
        im = image_reshape(im, (224, 224)) / 255.0
        image_array.append(im)
        label_array.append(label)

    label_array = encode_label(label_array)
    return np.array(image_array), np.array(label_array)
batch_size = 16
epoch_shuffle = True
weight_classes = True
epochs = 15

# Split train data up into 80% and 20% validation
train, validate = np.split(df.sample(frac=1), [int(.8*len(df))])
```

```
print("Training on:", len(train), "samples")
print("Validation on:", len(validate), "samples")


for e in range(epochs):
    print("Epoch: ", str(e+1) + "/" + str(epochs))
    if epoch_shuffle:
        train = train.sample(frac = 1)
    for it in range(int(np.ceil(len(train)/batch_size))):

        X_train, y_train = get_batch(train, it*batch_size, batch_size)

        model.train_on_batch(X_train, y_train)


model.save("Model.h5")
```

```
Training on: 16000 samples
Validation on: 4001 samples
Epoch:  1/15
Epoch:  2/15
Epoch:  3/15
Epoch:  4/15
Epoch:  5/15
Epoch:  6/15
Epoch:  7/15
Epoch:  8/15
Epoch:  9/15
Epoch:  10/15
Epoch:  11/15
Epoch:  12/15
Epoch:  13/15
Epoch:  14/15
Epoch:  15/15
```

الآن قمنا بتدريب النموذج بنجاح. الخطوة التالية هي اختبار النموذج، دعنا نرى كيف يمكننا اختبار نموذج اكتشاف المعالم لدينا:

```
### Test on training set
batch_size = 16

errors = 0
good_preds = []
bad_preds = []
```

```
for it in range(int(np.ceil(len(validate)/batch_size))):

    X_train, y_train = get_batch(validate, it*batch_size, batch_size)

    result = model.predict(X_train)
    cla = np.argmax(result, axis=1)
    for idx, res in enumerate(result):
        print("Class:", cla[idx], "- Confidence:",
np.round(res[cla[idx]],2), "- GT:", y_train[idx])
        if cla[idx] != y_train[idx]:
            errors = errors + 1
            bad_preds.append([batch_size*it + idx, cla[idx],
res[cla[idx]]])
        else:
            good_preds.append([batch_size*it + idx, cla[idx],
res[cla[idx]]])

print("Errors: ", errors, "Acc:", np.round(100*(len(validate)-
errors)/len(validate),2))

#Good predictions
good_preds = np.array(good_preds)
good_preds = np.array(sorted(good_preds, key = lambda x: x[2],
reverse=True))

fig=plt.figure(figsize=(16, 16))
for i in range(1,6):
    n = int(good_preds[i,0])
    img, lbl = get_image_from_number(n, validate)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    fig.add_subplot(1, 5, i)
    plt.imshow(img)
    lbl2 = np.array(int(good_preds[i,1])).reshape(1,1)
    sample_cnt = list(df.landmark_id).count(lbl)
    plt.title("Label: " + str(lbl) + "\nClassified as: " +
str(decode_label(lbl2)) + "\nSamples in class " + str(lbl) + ": " +
str(sample_cnt))
```

```
    plt.axis('off')
plt.show()
```

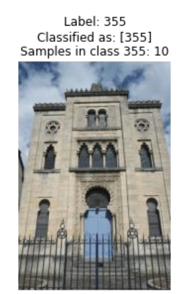
Label: 1924
Classified as: [1924]
Samples in class 1924: 944



Label: 27
Classified as: [27]
Samples in class 27: 504



Label: 355
Classified as: [355]
Samples in class 355: 10



Label: 1924
Classified as: [1924]
Samples in class 1924: 944



Label: 1924
Classified as: [1924]
Samples in class 1924: 944



كما ترىفي الصور أعلاهفي المخرجات، فقد تم تصنيفها حسب تسمياتها وفئاتها. آمل أن تكون قد أحببت هذه المقالة حول مشروع التعلم الآلي على Google Landmark Detection باستخدام لغة برمجة بايثون.

5) المتحدث الآلي مع التعلم الالي وبايثون Chatbot with Machine Learning and Python

إذا كنت مهتمًا بتطوير بوت محادثة (المتحدث الآلي) Chatbot، فقد تجد أن هناك العديد من أطر تطوير البوتات القوية والأدوات والأنظمة الأساسية التي يمكن استخدامها لتنفيذ برامج بوتات الدردشة الذكية.في هذه المقالة، سوف أطلعك على كيفية إنشاء Chatbot باستخدام بايثون والتعلم الآلي.

كيف يعمل Chatbot؟

نظرًا لأننا سنطور بوت محادثة Chatbot مع بايثون باستخدام التعلم الآلي، فنحن بحاجة إلى بعض البيانات لتدريب نموذجنا. لكننا لن نقوم بجمع أو تنزيل مجموعة بيانات كبيرة لأن هذا مجرد برنامج محادثة. يمكننا فقط إنشاء مجموعة البيانات الخاصة بنا لتدريب النموذج.
لإنشاء مجموعة البيانات هذه لإنشاء بوت محادثة باستخدام بايثون، نحتاج إلى فهم النوايا التي سنقوم بتدريبها. "النية intention " هي نية المستخدم للتفاعل مع روبوت محادثة أو النية وراء كل رسالة يتلقاها روبوت المحادثة من مستخدم معين.
لذلك، من المهم أن تفهم النوايا الحسنة لبرنامج chatbot الخاص بك اعتمادًا على المجال الذي ستعمل معه. فلماذا يحتاج إلى تحديد هذه النوايا؟ هذه نقطة مهمة للغاية يجب فهمها.
للإجابة على الأسئلة التي يطرحها المستخدمون وأداء العديد من المهام الأخرى لمواصلة المحادثات مع المستخدمين، يحتاج برنامج الدردشة الآلي حقًا إلى فهم ما يقوله المستخدمون أو لديهم نيةفي القيام به. هذا هو السببفي أن برنامج الدردشة الآلي الخاص بك يجب أن يفهم النوايا الكامنة وراء رسائل المستخدمين.
كيف يمكنك جعل chatbot الخاص بك يفهم النوايا حتى يشعر المستخدمون أنهم يعرفون ما يريدون ويقدمون إجابات دقيقة؟ تتمثل الإستراتيجية هنافي تعيين نوايا مختلفة وإنشاء عينات تدريب لهذه النوايا وتدريب نموذج chatbot الخاص بك باستخدام بيانات التدريب النموذجية هذه كبيانات تدريب نموذجية (X) والنواياڤي فئات تدريب نموذجية (Y).

إنشاء Chatbot باستخدام بايثون والتعلم الآلي

لإنشاء بوت محادثة باستخدام بايثون وتعلم آلي، تحتاج إلى تثبيت بعض الحزم. جميع الحزم التي تحتاج إلى تثبيتها لإنشاء بوت محادثة مع التعلم الآلي باستخدام لغة برمجة بايثون مذكورة أدناه:

- tensorflow==2.3.1
- nltk==3.5
- colorama==0.4.3
- numpy==1.18.5
- scikit_learn==0.23.2
- Flask==1.1.2

تحديد نوايا Chatbot

نحتاج الآن إلى تحديد بعض النوايا البسيطة ومجموعة من الرسائل التي تطابق تلك النوايا وأيضًا تعيين بعض الردود بناءً على كل فئة نوايا. سأقوم بإنشاء ملف JSON باسم "intents.json" بما في ذلك هذه البيانات على النحو التالي:

```
{"intents": [
    {"tag": "greeting",
     "patterns": ["Hi", "Hey", "Is anyone there?", "Hello", "Hay"],
     "responses": ["Hello", "Hi", "Hi there"]
    },
    {"tag": "goodbye",
     "patterns": ["Bye", "See you later", "Goodbye"],
     "responses": ["See you later", "Have a nice day", "Bye! Come back
again"]
    },
    {"tag": "thanks",
     "patterns": ["Thanks", "Thank you", "That's helpful", "Thanks for
the help"],
     "responses": ["Happy to help!", "Any time!", "My pleasure", "You're
most welcome!"]
    },
    {"tag": "about",
     "patterns": ["Who are you?", "What are you?", "Who you are?" ],
     "responses": ["I.m Joana, your bot assistant", "I'm Joana, an
Artificial Intelligent bot"]
    },
    {"tag": "name",
    "patterns": ["what is your name", "what should I call you", "whats
your name?"],
    "responses": ["You can call me Joana.", "I'm Joana!", "Just call me
as Joana"]
    },
    {"tag": "help",
    "patterns": ["Could you help me?", "give me a hand please", "Can you
help?", "What can you do for me?", "I need a support", "I need a help",
"support me please"],
    "responses": ["Tell me how can assist you", "Tell me your problem to
assist you", "Yes Sure, How can I support you"]
```

```
    },
    {"tag": "createaccount",
    "patterns": ["I need to create a new account", "how to open a new
account", "I want to create an account", "can you create an account for
me", "how to open a new account"],
    "responses": ["You can just easily create a new account from our web
site", "Just go to our web site and follow the guidelines to create a
new account"]
    },
    {"tag": "complaint",
    "patterns": ["have a complaint", "I want to raise a complaint",
"there is a complaint about a service"],
    "responses": ["Please provide us your complaint in order to assist
you", "Please mention your complaint, we will reach you and sorry for
any inconvenience caused"]
    }
]
}
```

تحضير البيانات

تتمثل الخطوة الثانية من هذه المهمة لإنشاء بوت محادثة باستخدام بايثون والتعلم الآلي في إعداد البيانات لتدريب بوت المحادثة الخاص بنا. سأبدأ هذه الخطوة عن طريق استيراد المكتبات والحزم الضرورية:

```
import json
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding,
GlobalAveragePooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
```

الآن سأقرأ ملف JSON وأعالج الملفات المطلوبة:

```
with open('intents.json') as file:
    data = json.load(file)

training_sentences = []
```

```
training_labels = []
labels = []
responses = []


for intent in data['intents']:
    for pattern in intent['patterns']:
        training_sentences.append(pattern)
        training_labels.append(intent['tag'])
    responses.append(intent['responses'])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])

num_classes = len(labels)
```

نحتاج الآن إلى استخدام طريقة ترميز التسمية label encoder التي توفرها مكتبة Scikit-Learnفي بايثون:

```
lbl_encoder = LabelEncoder()
lbl_encoder.fit(training_labels)
training_labels = lbl_encoder.transform(training_labels)
```

الترميز Tokenization

نحتاج الآن إلى ترميز البيانات باستخدام طريقة Tokenization لإنشاء بوت محادثة باستخدام بايثون والتعلم الآلي:

```
vocab_size = 1000
embedding_dim = 16
max_len = 20
oov_token = "<OOV>"

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(training_sentences)
padded_sequences = pad_sequences(sequences, truncating='post',
maxlen=max_len)
```

تدريب شبكة عصبية

الآن الخطوة التالية والأكثر أهميةفي عملية بناء بوت محادثة باستخدام بايثون والتعلم الآلي هي تدريب شبكة عصبية. الآن، سأتدرب وأنشئ شبكة عصبية لتدريب بوت المحادثة الخاص بنا:

```
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=max_len))
model.add(GlobalAveragePooling1D())
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

model.summary()
epochs = 500
history = model.fit(padded_sequences, np.array(training_labels),
epochs=epochs)
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 20, 16)	16000
global_average_pooling1d (Gl	(None, 16)	0
dense (Dense)	(None, 16)	272
dense_1 (Dense)	(None, 16)	272
dense_2 (Dense)	(None, 8)	136

```
Total params: 16,680
Trainable params: 16,680
Non-trainable params: 0
```

حفظ الشبكة العصبية:

لقد قمنا بتدريب النموذج، ولكن قبل أن نذهب إلى أبعد من ذلكفي عملية إنشاء بوت محادثة باستخدام بايثون والتعلم الآلي، دعنا نحفظ النموذج حتى نتمكن من استخدام هذه الشبكة العصبيةفي المستقبل أيضًا:

```
# to save the trained model
model.save("chat_model")

import pickle

# to save the fitted tokenizer
```

```
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)

# to save the fitted label encoder
with open('label_encoder.pickle', 'wb') as ecn_file:
    pickle.dump(lbl_encoder, ecn_file,
protocol=pickle.HIGHEST_PROTOCOL)
```

بناء Chatbot باستخدام بايثون ونموذج التعلم الآلي المدرب

سأقوم الآن بتنفيذ وظيفة الدردشة للتفاعل مع مستخدم حقيقي. عندما يتم استلام الرسالة من المستخدم، يقوم بوت المحادثة بحساب التشابه بين تسلسل النص الجديد وبيانات التدريب. مع الأخذفي الاعتبار درجات الثقة التي تم الحصول عليها لكل فئة، فإنه يصنف رسالة المستخدم وفقًا لنية مع أعلى درجة ثقة:

```
mport json
import numpy as np
from tensorflow import keras
from sklearn.preprocessing import LabelEncoder

import colorama
colorama.init()
from colorama import Fore, Style, Back

import random
import pickle

with open("intents.json") as file:
    data = json.load(file)


def chat():
    # load trained model
    model = keras.models.load_model('chat_model')

    # load tokenizer object
    with open('tokenizer.pickle', 'rb') as handle:
        tokenizer = pickle.load(handle)
```

```
# load label encoder object
    with open('label_encoder.pickle', 'rb') as enc:
        lbl_encoder = pickle.load(enc)

    # parameters
    max_len = 20

    while True:
        print(Fore.LIGHTBLUE_EX + "User: " + Style.RESET_ALL, end="")
        inp = input()
        if inp.lower() == "quit":
            break
 result = model.predict(keras.preprocessing.sequence.pad_sequences
(tokenizer.texts_to_sequences([inp]),
for i in data['intents']:
            if i['tag'] == tag:
                print(Fore.GREEN + "ChatBot:" + Style.RESET_ALL ,
np.random.choice(i['responses']))

        # print(Fore.GREEN + "ChatBot:" +
Style.RESET_ALL,random.choice(responses))

print(Fore.YELLOW + "Start messaging with the bot (type quit to stop)!"
+ Style.RESET_ALL)
chat()
```

```
Start messaging with the bot (type quit to stop)!
User:
```

هذه هي الطريقة التي يمكننا بها إنشاء chatbot باستخدام بايثون والتعلم الآلي. آمل أن تكون قد أحببت هذه المقالة حول كيفية إنشاء Chatbot باستخدام بايثون والتعلم الآلي.

6) منشئ العنوان مع التعلم الآلي Title Generator with Machine Learning

في هذه المقالة، سأستخدم مجموعة بيانات مقاطع الفيديو الشائعة على YouTube ولغة برمجة بايثون لتدريب نموذج للغة إنشاء النص باستخدام التعلم الآلي، والذي سيتم استخدامه لمهمة إنشاء عنوان لمقاطع فيديو youtube أو حتى لمدوناتك.

منشئ العنوان Title generator مهمة معالجة لغة طبيعية natural language processing وهي قضية مركزية للعديد من التعلم الآلي، بمافي ذلك توليف النص text synthesis، والكلام إلى نص speech to text، وأنظمة المحادثة conversational systems.

لبناء نموذج لمهمة منشئ العنوان أو منشئ النص، يجب تدريب النموذج على معرفة احتمالية حدوث كلمة، باستخدام الكلمات التي ظهرت بالفعل في التسلسل كسياق.

منشئ العنوان مع التعلم الآلي

سأبدأ هذه المهمة لإنشاء مولد عنوان باستخدام بايثون والتعلم الآلي عن طريق استيراد المكتبات وقراءة مجموعات البيانات. يمكن تنزيل مجموعات البيانات التي أستخدمهافي هذه المهمة من هنا:

```
import pandas as pd
import string
import numpy as np
import json

from keras.preprocessing.sequence import pad_sequences
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping
from keras.models import Sequential
import keras.utils as ku

import tensorflow as tf
tf.random.set_seed(2)
from numpy.random import seed
seed(1)

#load all the datasets
df1 = pd.read_csv('USvideos.csv')
df2 = pd.read_csv('CAvideos.csv')
```

```
df3 = pd.read_csv('GBvideos.csv')

#load the datasets containing the category names
data1 = json.load(open('US_category_id.json'))
data2 = json.load(open('CA_category_id.json'))
data3 = json.load(open('GB_category_id.json'))
```

نحتاج الآن إلى معالجة بياناتنا حتى نتمكن من استخدام هذه البيانات لتدريب نموذج التعلم الآلي الخاص بنا على مهمة منشئ العنوان. فيما يلي جميع خطوات تنظيف البيانات ومعالجتها التي نحتاج إلى اتباعها:

```
def category_extractor(data):
    i_d = [data['items'][i]['id'] for i in range(len(data['items']))]
    title = [data['items'][i]['snippet']["title"] for i in
range(len(data['items']))]
    i_d = list(map(int, i_d))
    category = zip(i_d, title)
    category = dict(category)
    return category

#create a new category column by mapping the category names to their
id
df1['category_title'] =
df1['category_id'].map(category_extractor(data1))
df2['category_title'] =
df2['category_id'].map(category_extractor(data2))
df3['category_title'] =
df3['category_id'].map(category_extractor(data3))

#join the dataframes
df = pd.concat([df1, df2, df3], ignore_index=True)

#drop rows based on duplicate videos
df = df.drop_duplicates('video_id')

#collect only titles of entertainment videos
#feel free to use any category of video that you want
entertainment = df[df['category_title'] == 'Entertainment']['title']
entertainment = entertainment.tolist()
```

```
#remove punctuations and convert text to lowercase
def clean_text(text):
    text = ''.join(e for e in text if e not in
string.punctuation).lower()

    text = text.encode('utf8').decode('ascii', 'ignore')
    return text

corpus = [clean_text(e) for e in entertainment]
```

توليد التسلسلات

تتطلب مهام معالجة اللغة الطبيعية إدخال بياناتفي شكل سلسلة من الرموز المميزة tokens.
الخطوة الأولى بعد تنقية البيانات هي إنشاء سلسلة من الرموز المميزة n-gram.
n-gram هو تسلسل مجاور لعدد n من العناصر لعينة معينة من النص أو المجموعة الصوتية vocal corpus. يمكن أن تكون العناصر كلمات أو مقاطع لفظية أو مقاطع صوتية أو أحرف أو أزواج أساسية.في هذه الحالة، فإن n-grams هي سلسلة من الكلماتفي مجموعة من العناوين.
الترميز Tokenization هو عملية استخراج الرموز من المجموعة corpus:

```
tokenizer = Tokenizer()
def get_sequence_of_tokens(corpus):
  #get tokens
  tokenizer.fit_on_texts(corpus)
  total_words = len(tokenizer.word_index) + 1

  #convert to sequence of tokens
  input_sequences = []
  for line in corpus:
  token_list = tokenizer.texts_to_sequences([line])[0]
  for i in range(1, len(token_list)):
  n_gram_sequence = token_list[:i+1]
  input_sequences.append(n_gram_sequence)

  return input_sequences, total_words
inp_sequences, total_words = get_sequence_of_tokens(corpus)
```

نظرًا لأن التسلسلات يمكن أن تكون ذات أطوال متغيرة، يجب أن تكون أطوال التسلسل متساوية.
عند استخدام الشبكات العصبية، عادة ما نقوم بتغذية مدخلاتفي الشبكة أثناء انتظار الإخراج.

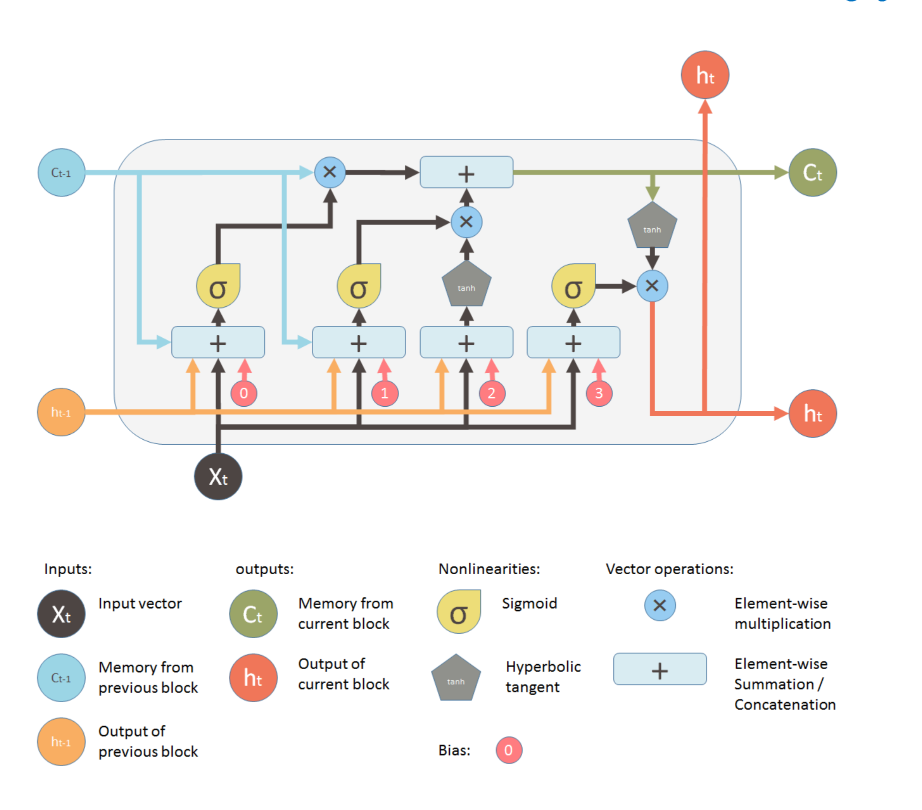
من الناحية العملية، من الأفضل معالجة البيانات على دفعات بدلاً من معالجة البيانات دفعة واحدة.

يتم ذلك باستخدام المصفوفات [حجم الدُفعة × طول التسلسل]، حيث يتوافق طول التسلسل مع أطول تسلسل.في هذه الحالة، نقوم بملء التسلسلات برمز (عادة 0) ليناسب حجم المصفوفة. تسمى هذه العملية لملء التسلسلات بالرموز المميزة الملء filling. لإدخال البياناتفي نموذج تدريب، أحتاج إلى إنشاء تنبؤات predictors وتسميات labels.

سأقوم بإنشاء متواليات من n-gram كمتنبئات والكلمة التالية من n-gram كتسمية:

```
def generate_padded_sequences(input_sequences):
  max_sequence_len = max([len(x) for x in input_sequences])
  input_sequences = np.array(pad_sequences(input_sequences,
maxlen=max_sequence_len, padding=’pre’))
  predictors, label = input_sequences[:,:-1], input_sequences[:, -1]
  label = ku.to_categorical(label, num_classes = total_words)
  return predictors, label, max_sequence_len
predictors, label, max_sequence_len =
generate_padded_sequences(inp_sequences)
```

نموذج LSTM



في الشبكات العصبية المتكررة recurrent neural networks(RNN)، يتم نشر مخرجات التنشيط في كلا الاتجاهين، أي من المدخلات إلى المخرجات والمخرجات إلى المدخلات، على عكس الشبكات العصبية ذات التمثيل المباشر حيث تنتشر المخرجات d التنشيط في اتجاه واحد فقط. يؤدي هذا إلى إنشاء حلقات في بُنية الشبكة العصبية التي تعمل بمثابة "حالة ذاكرة memory state" للخلايا العصبية.

نتيجة لذلك، تحتفظ RNN بحالة من خلال مراحل الوقت أو "تتذكر remembers" ما تم تعلمه بمرور الوقت. لحالة الذاكرة مزاياها، ولكن لها أيضًا عيوبها. التدرج gradient الذي يختفي هو واحد منهم.

في هذه المشكلة، أثناء التعلم باستخدام عدد كبير من الطبقات، يصبح من الصعب حقًا على الشبكة تعلم وضبط معلمات الطبقات السابقة. لحل هذه المشكلة، تم تطوير نوع جديد من RNN؛ LSTM (ذاكرة طويلة قصيرة المدى Long Short Term Memory).

مولد العنوان مع نموذج LSTM

يحتوي نموذج LSTM على حالة إضافية (حالة الخلية) والتي تسمح أساسًا للشبكة بمعرفة ما يتم تخزينه في الحالة طويلة المدى، وما يجب حذفه وما يجب قراءته. يحتوي LSTM لهذا النموذج على أربع طبقات:

- **طبقة الإدخال Input layer:** تأخذ تسلسل الكلمات كمدخل.
- **طبقة LSTM:** تحسب الناتج باستخدام وحدات LSTM.
- **طبقة التسرب Dropout layer:** طبقة تنظيم لتجنب فرط التعلم overfitting.
- **طبقة الإخراج Output layer:** تحسب احتمالية الكلمة التالية المحتملة عند الإخراج.

سأستخدم الآن نموذج LSTM لبناء نموذج لمهمة منشئ العنوان Title Generator مع التعلم الآلي:

```
def create_model(max_sequence_len, total_words):
  input_len = max_sequence_len - 1
  model = Sequential()

  # Add Input Embedding Layer
  model.add(Embedding(total_words, 10, input_length=input_len))

  # Add Hidden Layer 1 - LSTM Layer
  model.add(LSTM(100))
  model.add(Dropout(0.1))

  # Add Output Layer
  model.add(Dense(total_words, activation='softmax'))
  model.compile(loss='categorical_crossentropy', optimizer='adam')
```

```
  return model
model = create_model(max_sequence_len, total_words)
model.fit(predictors, label, epochs=20, verbose=5)
```

مولد العنوان مع التعلم الآلي: اختبار النموذج

الآن وقد أصبح نموذج التعلم الآلي الخاص بنا لمنشئ العنوان جاهزًا وتم تدريبه باستخدام البيانات، فقد حان الوقت للتنبؤ بالعنوان بناءً على كلمة الإدخال. يتم ترميز كلمة الإدخال أولاً، ثم يكتمل التسلسل قبل أن يتم تمريرها إلى النموذج المدرب لإرجاع التسلسل المتوقع:

```
def generate_text(seed_text, next_words, model, max_sequence_len):
  for _ in range(next_words):
  token_list = tokenizer.texts_to_sequences([seed_text])[0]
  token_list = pad_sequences([token_list], maxlen=max_sequence_len-
1,  padding='pre')
  predicted = model.predict_classes(token_list, verbose=0)

  output_word = ""
  for word,index in tokenizer.word_index.items():
  if index == predicted:
  output_word = word
  break
  seed_text += " "+output_word
  return seed_text.title()
```

توليد العناوين

الآن بما أننا أنشأنا دالة لإنشاء العناوين، فلنختبر نموذج منشئ العنوان الخاص بنا:

```
print(generate_text("spiderman", 5, model, max_sequence_len))
```

```
Output:  Spiderman The Voice 2018 Blind Audition
```

أتمنى أن تكون قد أحببت هذه المقالة حول كيفية إنشاء نموذج منشئ العنوان باستخدام التعلم الآلي ولغة برمجة بايثون.

7) كشف التزييف العميق باستخدام بايثون Deepfake Detection with Python

في هذه المقالة، سوف آخذك إلى كشف التزييف العميق Deepfake Detection باستخدام بايثون والتعلم الآلي. لقد كتبت مؤخرًا مقالًا حول ما هو التزييف العميق Deepfake وكيف أنه خطير. إذا كنت لا تعرف ما هو Deepfake، فسأقترح عليك إلقاء نظرة سريعة على مقالتي السابقة هنا، قبل أن تتسخ يديك بمهمة اكتشاف التزييف العميق Deepfake باستخدام بايثون والتعلم الآلي.

كشف التزييف العميق باستخدام بايثون

كانت هناك العديد من التقارير عن مقاطع فيديو مزيفة لمشاهير أو سياسيين مشهورين. يصعب اكتشاف مقاطع الفيديو المزيفة هذه بالعين المجردة وأصبحت مشكلة رئيسيةفي المجتمع.

لقد تمت تجربة حتى الآن أن مقاطع فيديو Deepfake تنتشر بسهولة على منصات مثل Facebook وtwitter وyoutube وما إلى ذلك. نظرًا لأن هذه المنصات تعمل على حل هذه المشكلة، فإن Facebook يقوم باستثمار كبير (10 ملايين دولار) لإصلاح هذه المشكلة، وتعمل منصات أخرى مثل Twitter وGoogle أيضًا على حل هذه المشكلة.

وبالتالي، فإن اكتشاف التزييف العميق ليس بالمهمة السهلة.في هذه المقالة، سنرى كيفية التعرف على المنتجات المقلدة من الحقيقية. يتضمن تحليل مقاطع الفيديوفي إطار، واكتشاف الوجوه من مقاطع الفيديو الحقيقية والمزيفة، واقتصاص الوجوه وتحليلها.

كشف التزييف العميق أثناء العمل

دعنا الآن نرى كيف يمكننا كشف محتوى التزييف العميق Deepfake باستخدام بايثون والتعلم الآلي. سأبدأ بهذه المهمة عن طريق استيراد المكتبات الضرورية:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import pandas as pd
import glob2
import os, fnmatch
from pathlib import Path
# import mtcnn
from mtcnn.mtcnn import MTCNN
```

الآن، لنفترض أن لدينا نوعين من مقاطع الفيديو، أحدهما حقيقي والآخر مزيف نريد اكتشافه أيهما مزيف من بين الاثنين، ربما هذا هو ما نحتاج إلى القيام بهفي مهمة اكتشاف التزييف العميق Deepfake.

الآن، سوف أقوم بإنشاء دالة لمعالجة كلا مقاطع الفيديو:

```
def extract_multiple_videos(intput_filenames, image_path_infile):
    """Extract video files into sequence of images."""
i = 1  # Counter of first video
# Iterate file names:
    cap = cv2.VideoCapture('your_video_file_path.avi' or
intput_filenames)
if (cap.isOpened()== False):
        print("Error opening file")
# Keep iterating break
    while True:
        ret, frame = cap.read()  # Read frame from first video

        if ret:
            cv2.imwrite(os.path.join(image_path_infile , str(i) +
'.jpg'), frame)  # Write frame to JPEG file (1.jpg, 2.jpg, ...)
# you can uncomment this line if you want to view them.
#           cv2.imshow('frame', frame)  # Display frame for testing
            i += 1 # Advance file counter
        else:
            # Break the interal loop when res status is False.
            break
cv2.waitKey(50) #Wait 50msec
cap.release()
```

استدعاء الدالة

```
extract_multiple_videos(fake_video_name, fake_image_path_for_frame)
extract_multiple_videos(real_video_name, real_image_path_for_frame)
```

الآن بعد تشغيل الدالة، ستتمكن من قراءة مقاطع الفيديو ومعالجتها لمهمة اكتشاف التزييف العميق Deepfake. دعنا الآن نرى كيف يمكننا تحديد التزييف العميق من خلال مقارنة كل من مقاطع الفيديو:

```
from skimage import measure
def mse(imageA, imageB):
    # the 'Mean Squared Error' between the two images is the
    # sum of the squared difference between the two images;
    # NOTE: the two images must have the same dimension
    err = np.sum((imageA.astype("float") - imageB.astype("float")) **
2)
    err /= float(imageA.shape[0] * imageA.shape[1])
# return the MSE, the lower the error, the more "similar"
```

```
    # the two images are
    return err
def compare_images(imageA, imageB, title):
    # compute the mean squared error and structural similarity
    # index for the images
    m = mse(imageA, imageB)
    s = measure.compare_ssim(imageA, imageB)
    # setup the figure
    fig = plt.figure(title)
    plt.suptitle("MSE: %.2f, SSIM: %.2f" % (m, s))
    # show first image
    ax = fig.add_subplot(1, 2, 1)
    plt.imshow(imageA, cmap = plt.cm.gray)
    plt.axis("off")
    # show the second image
    ax = fig.add_subplot(1, 2, 2)
    plt.imshow(imageB, cmap = plt.cm.gray)
    plt.axis("off")
    # show the images
    plt.show()
```

في الكود أعلاه، نقارن الصور المستخرجة من الفيديو الأصلي والصورة المقابلة من مقاطع الفيديو المزيفة.في القسم الأخير من الكود، تحققت من وجود أي اختلافات بين الصورتين.

المخرجات:

Original



Modified



هذه هي الطريقة التي يمكننا بها الكشف عن ملفات التزييف العميق من خلال مقارنة الملفات الأصلية والمزيفة. أتمنى أن تكون قد أحببت هذه المقالة حول التزييف العميق Deepfake باستخدام بايثون والتعلم العميق.

8) تصنيف الجنسيات باستخدام التعلم الآلي Classify Nationalities with Machine Learning

في هذا المقال، سوف أطلعكم على كيفية تصنيف جنسيات الأشخاص باستخدام أسمائهم. سوف تفكرفي كيفية تصنيف الجنسيات باستخدام الأسماء فقط. هناك الكثير حول كيفية اللعب بالأسماء.

تصنيف الجنسيات

لنبدأ بمهمة التعلم الآلي هذه لتصنيف الجنسيات عن طريق استيراد الحزم الضرورية. سأصنف الجنسيات بناءً على الأسماء على أنها هندية أو غير هندية. لذلك، دعنا نستورد بعض الحزم ونبدأ بالمهمة:

```
from tensorflow import keras
import tensorflow as tf
import pandas as pd
import os
import re
```

الآن، دعنا نستورد مجموعات البيانات. يمكن تنزيل مجموعات البيانات التي أستخدمها هنافي هذه المقالة بسهولة من هنا. الآن بعد استيراد مجموعات البيانات، سأقوم بإعداد دالتين مساعدتين لتنظيف البيانات ومعالجة البيانات:

```
male_data = pd.read_csv(male.csv)
female_data = pd.read_csv(femaile.csv)
repl_list = ['s/o','d/o','w/o','/','&',',','-']
def clean_data(name):
        name = str(name).lower()
        name = (''.join(i for i in name if ord(i)<128)).strip()
        for repl in repl_list:
                name = name.replace(repl," ")
        if '@' in name:
                pos = name.find('@')
                name = name[:pos].strip()
        name = name.split(" ")
        name = " ".join([each.strip() for each in name])
        return name

def remove_records(merged_data):
        merged_data['delete'] = 0
        merged_data.loc[merged_data['name'].str.find('with') != -1,
```

```
'delete'] = 1
        merged_data.loc[merged_data['count_words']>=5,'delete']=1
        merged_data.loc[merged_data['count_words']==0,'delete']=1
        merged_data.loc[merged_data['name'].str.contains(r'\d') == True,
'delete']=1
        cleaned_data = merged_data[merged_data.delete==0]
        return cleaned_data
merged_data = pd.concat((male_data,female_data),axis=0)
merged_data['name'] = merged_data['name'].apply(clean_data)
merged_data['count_words'] = merged_data['name'].str.split().apply(len)

cleaned_data = remove_records(merged_data)
indian_cleaned_data = cleaned_data[['name','count_words']].drop_duplicates(
subset='name',keep='first')
indian_cleaned_data['label'] = 'indian'
len(indian_cleaned_data)
```

```
13754
```

بعد تحميل وإزالة الإدخالات الخاطئةفي البيانات، حصلنا على عدد قليل من السجلات تقريبا 13000.

بالنسبة للأسماء غير الهندية، هناك حزمة أنيقة تسمى Faker. هذا يولد أسماء من مناطق مختلفة:

```
from faker import Faker
fake = Faker('en_US')
fake.name()
```

```
'Brian Evans'
```

لقد أنشأنا نفس عدد الأسماء تقريبًا كما لدينافي مجموعة البيانات الهندية. ثم أزلنا عينات أطول من 5 كلمات. احتوت مجموعة البيانات الهندية على الكثير من الأسماء بأسماء أولى فقط. لذلك نحن بحاجة إلى جعل التوزيع العام لغير الهند متشابهًا أيضًا.

```
non_indian_data.head()
```

	name	count_words
0	sara gulbrandsen	2
1	kathryn villarreal	2
2	jennifer mccormick	2
3	james eaton	2
4	melissa bond	2

انتهى بنا المطاف بحوالي 14000 اسم غير هندي و13000 اسم هندي. لنقم الآن ببناء شبكة عصبية لتصنيف الجنسيات باستخدام الأسماء:

```
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
from keras.layers.embeddings import Embedding
from keras.utils import to_categorical
import numpy as np
from sklearn.preprocessing import LabelEncoder
from keras.callbacks import Callback
np.random.seed(42)
def char_encoded_representation(data,tokenizer,vocab_size,max_len):
  char_index_sentences = tokenizer.texts_to_sequences(data)
  sequences = [to_categorical(x, num_classes=vocab_size) for x in
char_index_sentences]
  X = sequence.pad_sequences(sequences, maxlen=max_len)
  return X

def build_model(hidden_units,max_len,vocab_size):
  model = Sequential()
  model.add(LSTM(hidden_units,input_shape=(max_len,vocab_size)))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
  print(model.summary())
  return model
```

```
model = build_model(100,max_len,vocab_size)
model.fit(X_train, y_train, epochs=20,
batch_size=64,callbacks=myCallback(X_test,y_test))
```

	names	predictions_lstm_char
0	lalitha	indian
1	tyson	non_indian
2	shailaja	indian
3	shyamala	indian
4	vishwanathan	indian
5	ramanujam	indian
6	conan	non_indian
7	kryslovsky	non_indian
8	ratnani	indian
9	diego	non_indian
10	kakoli	indian
11	shreyas	indian
12	brayden	non_indian
13	shanon	non_indian

هذه هي الطريقة التي يمكننا بها تصنيف الجنسيات بسهولة باستخدام التعلم الآلي. لم أقم بتضمين الكود الكامل والاستكشاف هنا، يمكنك إلقاء نظرة على الكود الكامل من هنا.

9) توقع أسعار السيارات مع التعلم الآلي Predict Car Prices with Machine Learning

في هذه المقالة، سوف أطلعك على كيفية تدريب نموذج يساعدنافي التنبؤ بأسعار السيارات باستخدام التعلم الآلي باستخدام PyTorch. مجموعة البيانات التي سأستخدمها هنا للتنبؤ بأسعار السيارات عبارة عن بيانات مجدولة مع أسعار السيارات المختلفة فيما يتعلق بالمتغيرات الأخرى، وتحتوي مجموعة البيانات على 258 صفاً و9 أعمدة، والمتغير الذي نريد توقعه هو سعر بيع السيارات.

ما هو PyTorch؟

PyTorch هي مكتبةفي بايثون توفر أدوات لبناء نماذج التعلم العميق. ما يفعله Pythonفي البرمجة يفعله PyTorch للتعلم العميق. بايثون هي لغة مرنة للغاية للبرمجة ومثل لغة بايثون، توفر مكتبة PyTorch أدوات مرنة للتعلم العميق. إذا كنت تتعلم التعلم العميق أو تتطلع إلى البدء به، فإن معرفة PyTorch ستساعدك كثيرًافي إنشاء نماذج التعلم العميق الخاصة بك.

توقع أسعار السيارات باستخدام PyTorch

الآن، لنبدأ بمهمة التعلم الآلي للتنبؤ بأسعار السيارات باستخدام PyTorch. سأبدأ باستيراد جميع المكتبات الضرورية التي نحتاجها لهذه المهمة. يمكن تنزيل مجموعة البيانات التي أستخدمهافي هذه المهمة بسهولة من هنا:

```
import torch
import jovian
import torch.nn as nn
import pandas as pd
import matplotlib.pyplot as plt
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset, random_split
```

الآن، دعنا نقرأ البيانات:

```
DATA_FILENAME = "car_data.csv"
dataframe_raw = pd.read_csv (DATA_FILENAME)
dataframe_raw.head()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

يمكنك رؤية شكل البيانات، ولكن قبل استخدامها، نحتاج إلى تخصيصها وفرز الأسهم وإزالة الأعمدة التي لا تساعدفي التنبؤ، وهنا نقوم بإسقاط أسماء السيارات، وللقيام بهذا التخصيص، نستخدم الدالة التالية:

```
your_name = "Aman Kharwal" # at least 5 characters
def customize_dataset(dataframe_raw, rand_str):
    dataframe = dataframe_raw.copy(deep=True)
    # drop some rows
    dataframe = dataframe.sample(int(0.95*len(dataframe)),
random_state=int(ord(rand_str[0])))
    # scale input
    dataframe.Year = dataframe.Year * ord(rand_str[1])/100.
    # scale target
    dataframe.Selling_Price = dataframe.Selling_Price *
ord(rand_str[2])/100.
    # drop column
    if ord(rand_str[3]) % 2 == 1:
        dataframe = dataframe.drop(['Car_Name'], axis=1)
    return dataframe

dataframe = customize_dataset(dataframe_raw, your_name)
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
296	1955.52	10.3550	11.60	33988	Diesel	Dealer	Manual	0
233	1952.61	4.2510	5.70	53000	Diesel	Dealer	Manual	0
15	1955.52	8.4475	10.79	43000	Diesel	Dealer	Manual	0
289	1955.52	11.0199	13.60	10980	Petrol	Dealer	Manual	0
155	1956.49	0.5232	0.51	4300	Petrol	Individual	Automatic	0

```
dataframe.head()
```

في هذه الدالة أعلاه كما نرى أنها تحتاج إلى كلمة لاستخدامها كسلسلة عشوائية لفرز البيانات بشكل عشوائي، استخدمت اسمي كسلسلة عشوائية. بعد ذلك يمكننا استخدام مجموعة البيانات

المخصصة، للتبسيط يمكننا إنشاء متغيرات تحتوي على عدد الصفوف والأعمدة والمتغيرات التي تحتوي على أعمدة رقمية أو فئوية أو ناتجة:

```
input_cols = ["Year","Present_Price","Kms_Driven","Owner"]
categorical_cols = ["Fuel_Type","Seller_Type","Transmission"]
output_cols = ["Selling_Price"]
```

تحضير البيانات

كما هو مذكورفي بداية المقال، سأستخدم PyTorch للتنبؤ بأسعار السيارات باستخدام التعلم الآلي، لذلك لاستخدام البيانات للتدريب نحتاج إلى تحويلها من إطار البيانات إلى PyTorch Tensors، فإن الخطوة الأولى هي التحويل إلى مصفوفات NumPy:

```
def dataframe_to_arrays(dataframe):
    # Make a copy of the original dataframe
    dataframe1 = dataframe.copy(deep=True)
    # Convert non-numeric categorical columns to numbers
    for col in categorical_cols:
        dataframe1[col] =
dataframe1[col].astype('category').cat.codes
    # Extract input & outupts as numpy arrays
    inputs_array = dataframe1[input_cols].to_numpy()
    targets_array = dataframe1[output_cols].to_numpy()
    return inputs_array, targets_array

inputs_array, targets_array = dataframe_to_arrays(dataframe)
inputs_array, targets_array
```

تقوم الدالة المذكورة أعلاه بتحويل أعمدة الإدخال والإخراج إلى مصفوفات NumPy، للتحقق من إمكانية عرض النتيجة وكما يمكنك رؤية كيفية تحويل البيانات إلى مصفوفات. الآن بوجود هذه المصفوفات، يمكننا تحويلها إلى موترات PyTorch، واستخدام تلك الموترات لإنشاء مجموعة بيانات متغيرة تحتوي عليها:

```
inputs = torch.Tensor(inputs_array)
targets = torch.Tensor(targets_array)

dataset = TensorDataset(inputs, targets)
train_ds, val_ds = random_split(dataset, [228, 57])
batch_size = 128

train_loader = DataLoader(train_ds, batch_size, shuffle=True)
val_loader = DataLoader(val_ds, batch_size)
```

إنشاء نموذج PyTorch

الآن، سأقوم بإنشاء نموذج انحدار خطي باستخدام PyTorch للتنبؤ بأسعار السيارات:

```
input_size = len(input_cols)
output_size = len(output_cols)

class CarsModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(input_size, output_size)
# fill this (hint: use input_size & output_size defined above)

    def forward(self, xb):
        out = self.linear(xb)                               # fill this
        return out

    def training_step(self, batch):
        inputs, targets = batch
        # Generate predictions
        out = self(inputs)
        # Calcuate loss
        loss = F.l1_loss(out, targets)                      # fill this
        return loss

    def validation_step(self, batch):
        inputs, targets = batch
        # Generate predictions
        out = self(inputs)
        # Calculate loss
        loss = F.l1_loss(out, targets)                        # fill this
        return {'val_loss': loss.detach()}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()   # Combine
losses
        return {'val_loss': epoch_loss.item()}
```

```
    def epoch_end(self, epoch, result, num_epochs):
        # Print result every 20th epoch
        if (epoch+1) % 20 == 0 or epoch == num_epochs-1:
            print("Epoch [{}], val_loss: {:.4f}".format(epoch+1,
result['val_loss']))

model = CarsModel()

list(model.parameters())
```

في هذه الدالة أعلاه، استخدمت الدالة nn.Linear التي ستسمح لنا باستخدام الانحدار الخطي حتى نتمكن الآن من حساب التنبؤات والخسارة باستخدام دالة F.l1_loss. يمكن لدالة F.l1_loss أن ترى تحيزًا لمعامل الوزن واحد، مع هذا النموذج سوف نحصل على التنبؤات، ولكن لا يزال يتعين علينا الخضوع للتدريب.

تدريب النموذج للتنبؤ بأسعار السيارات

نحتاج الآن إلى تقييم الخسارة ومعرفة مقدارها، وبعد القيام بالتدريب، انظر إلى أي مدى تتناقص الخسارة مع التدريب:

```
# Eval algorithm
def evaluate(model, val_loader):
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

# Fitting algorithm
def fit(epochs, lr, model, train_loader, val_loader,
opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):
        # Training Phase
        for batch in train_loader:
            loss = model.training_step(batch)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        # Validation phase
        result = evaluate(model, val_loader)
        model.epoch_end(epoch, result, epochs)
        history.append(result)
```

```
    return history

# Check the initial value that val_loss have
result = evaluate(model, val_loader)
print(result)
```

```
{'val_loss': 2300.039306640625}
```

```
# Start with the Fitting
epochs = 90
lr = 1e-8
history1 = fit(epochs, lr, model, train_loader, val_loader)
```

```
Epoch [20], val_loss: 1692.0131
Epoch [40], val_loss: 1119.7253
Epoch [60], val_loss: 638.9708
Epoch [80], val_loss: 357.3529
Epoch [90], val_loss: 317.1693
```

```
# Train repeatdly until have a 'good' val_loss
epochs = 20
lr = 1e-9
history1 = fit(epochs, lr, model, train_loader, val_loader)
```

```
Epoch [20], val_loss: 7.9774
```

كما ترون، من أجل التقييم، يتم استخدام دوال النموذج الملائم، للقيام بالتدريب، نستخدم دوال التحسين،في هذه الحالة على وجه التحديد تحسين SGD، باستخدام محمل التدريب يتم حساب الخسارة والتدرجات، لتحسينها بعد ذلك وتقييم نتيجة كل تكرار لرؤية الخسارة.

استخدام النموذج للتنبؤ بأسعار السيارات

أخيرًا، نحتاج إلى اختبار النموذج ببيانات محددة، للتنبؤ بضرورة استخدام المدخلات التي ستكون قيم الإدخال التي نراهافي مجموعة البيانات، والنموذج هو نموذج Cars الذي نقوم به، من أجل المرورفي النموذج ضروري للتسطيح، لذلك مع كل هذا توقع أسعار البيع:

```
# Prediction Algorithm
def predict_single(input, target, model):
    inputs = input.unsqueeze(0)
    predictions = model(inputs)                # fill this
    prediction = predictions[0].detach()
```

```
    print("Input:", input)
    print("Target:", target)
    print("Prediction:", prediction)

# Testing the model with some samples
input, target = val_ds[0]
predict_single(input, target, model)
```

```
Input: tensor([1.9565e+03, 1.7800e+00, 4.0000e+03, 0.0000e+00])
Target: tensor([1.7985])
Prediction: tensor([1.4945])
```

كما ترى، فإن التنبؤات قريبة جداً من الهدف المتوقع، وليست دقيقة ولكنها مشابهة لما هو متوقع. مع هذا الآن يمكنك اختبار نتائج مختلفة ومعرفة مدى جودة النموذج:

```
input, target = val_ds[10]
predict_single(input, target, model)
```

```
Input: tensor([1.9555e+03, 8.4000e+00, 1.2000e+04, 0.0000e+00])
Target: tensor([6.9760])
Prediction: tensor([-0.4069])
```

آمل أن تكون قد أحببت هذه المقالة حول كيفية التنبؤ بأسعار السيارات باستخدام التعلم الآلي باستخدام نموذج الانحدار الخطي المدرب باستخدام PyTorch.

10) توقع كفاءة الوقود باستخدام التعلم الآلي Predict Fuel Efficiency with Machine Learning

في هذه الأنواع من مشاكل التعلم الآلي للتنبؤ بكفاءة الوقود، نهدف إلى التنبؤ بمخرجات القيمة المستمرة، مثل السعر أو الاحتمالية.في هذه المقالة، سأطلعك على كيفية التنبؤ بكفاءة استهلاك الوقود من خلال التعلم الآلي.

توقع كفاءة الوقود

سأستخدم هنا إحدى مجموعات البيانات الشهيرة بين ممارسي التعلم الآلي، مجموعة بيانات Auto MPG لإنشاء نموذج للتنبؤ بكفاءة استهلاك الوقود للمركبات في أواخر السبعينيات وأوائل الثمانينيات. للقيام بذلك، سنزود النموذج بوصف للعديد من السيارات من هذه الفترة. يتضمن هذا الوصف سمات مثل الأسطوانات cylinders والإزاحة displacement والقدرة الحصانية horsepower والوزن weight.

دعنا نستورد المكتبات الضرورية للبدء بهذه المهمة:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

import tensorflow as tf
from tensorflow import keras

from tensorflow.keras import layers
```

الآن، الشيء التالي الذي يجب فعله هو تنزيل مجموعة البيانات. يمكنك بسهولة تنزيل مجموعة البيانات من هنا. الآن، دعنا نستورد البيانات باستخدام حزمة pandas:

```
column_names =
['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
dataset = pd.read_csv("auto.csv", names=column_names,
                      na_values = "?", comment='\t',

                      sep=" ", skipinitialspace=True)
```

عمود "origin"في مجموعة البيانات فئوي categorical، لذا للمضي قدمًا نحتاج إلى استخدام بعض الترميز الساخن one-hot encoding:

```
origin = dataset.pop('Origin')
dataset['USA'] = (origin == 1)*1.0
dataset['Europe'] = (origin == 2)*1.0

dataset['Japan'] = (origin == 3)*1.0
```

الآن، دعنا نقسم البيانات إلى مجموعات تدريب واختبار:

```
train_dataset = dataset.sample(frac=0.8,random_state=0)

test_dataset = dataset.drop(train_dataset.index)
```

قبل التدريب والاختبار للتنبؤ بكفاءة استهلاك الوقود من خلال التعلم الآلي، دعنا نتخيل البيانات باستخدام طريقة رسم ازواج seaborn:

```
sns.pairplot(train_dataset[["MPG", "Cylinders", "Displacement",
"Weight"]], diag_kind="kde")
```

الآن، سأفصل القيم المستهدفة عن الميزات الموجودةفي مجموعة البيانات. هذه التسمية هي تلك الميزة التي سأستخدمها لتدريب النموذج على التنبؤ بكفاءة الوقود:

```
train_labels = train_dataset.pop('MPG')
test_labels = test_dataset.pop('MPG')
```

تسوية البيانات

يوصى بتوحيد standardize الميزات التي تستخدم مقاييس ونطاقات مختلفة. على الرغم من أن النموذج يمكن أن يتقارب دون توحيد الميزات، إلا أن هذا يجعل التعلم أكثر صعوبة ويجعل النموذج الناتج يعتمد على اختيار الوحدات المستخدمةفي الإدخال. نحتاج إلى القيام بذلك لعرض مجموعة بيانات الاختبارفي نفس التوزيع الذي تم تدريب النموذج عليه:

```
def norm(x):
  return (x - train_stats['mean']) / train_stats['std']
normed_train_data = norm(train_dataset)
normed_test_data = norm(test_dataset)
```

ملاحظة: يجب تطبيق الإحصائيات المستخدمة لتسوية المدخلات هنا (المتوسط والانحراف المعياري) على جميع البيانات الأخرى المقدمة إلى النموذج، باستخدام الترميز الواحد الساخن الذي قمنا به سابقًا. يتضمن ذلك مجموعة الاختبار بالإضافة إلى البيانات المباشرة عند استخدام النموذجفي الإنتاج.

بناء النموذج

دعونا نبني نموذجنا. هنا، سأستخدم API التسلسلي مع طبقتين مخفيتين وطبقة إخراج واحدة ستعيد قيمة واحدة. يتم تغليف خطوات إنشاء النموذجفي دالة build_model، نظرًا لأننا سننشئ نموذجًا ثانيًا لاحقًا:

```
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation=tf.nn.relu,
input_shape=[len(train_dataset.keys())]),
    layers.Dense(64, activation=tf.nn.relu),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mean_squared_error',
                optimizer=optimizer,
                metrics=['mean_absolute_error', 'mean_squared_error'])
  return model
model = build_model()
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                640
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 4,865
Trainable params: 4,865
Non-trainable params: 0
_________________________________________________________________
```

الآن، قبل تدريب النموذج للتنبؤ بكفاءة الوقود، دعنا نضع هذا النموذج في العينات العشر الأولى:

```
example_batch = normed_train_data[:10]
example_result = model.predict(example_batch)
```

```
example_result
```

```
array([[-0.12670723],
       [-0.03443428],
       [ 0.3062502 ],
       [ 0.3065169 ],
       [ 0.36841604],
       [ 0.02191051],
       [ 0.3674207 ],
       [ 0.10561748],
       [ 0.00638346],
       [-0.00226454]], dtype=float32)
```

تدريب النموذج للتنبؤ بكفاءة الوقود

الآن، دعنا ندرب النموذج على التنبؤ بكفاءة الوقود:

```
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

EPOCHS = 1000

history = model.fit(
  normed_train_data, train_labels,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])
```

الآن، دعونا نرسم تدريب النموذج:

```
def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Abs Error [MPG]')
  plt.plot(hist['epoch'], hist['mean_absolute_error'],
           label='Train Error')
```

```
  plt.plot(hist['epoch'], hist['val_mean_absolute_error'],
           label = 'Val Error')
  plt.ylim([0,5])
  plt.legend()

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Square Error [$MPG^2$]')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
           label = 'Val Error')
  plt.ylim([0,20])
  plt.legend()
  plt.show()
plot_history(history)
```

يمثل هذا الرسم البياني أدناه تحسنًا طفيفًا أو حتى تدهورًافي خطأ التحقق بعد حوالي 100 حقبة. الآن، دعنا نحدِّث طريقة model.fit لإيقاف التدريب عندما لا تتحسن نتيجة التحقق. سنستخدم التوقف المبكر EarlyStopping الذي يختبر حالة التدريب لكل حقبة. إذا مر عدد محدد من الحقبات دون إظهار تحسن، فقم بإيقاف التدريب تلقائيًا:

```
model = build_model()

# The patience parameter is the amount of epochs to check for
improvement
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10)

history = model.fit(normed_train_data, train_labels, epochs=EPOCHS,
                    validation_split = 0.2, verbose=0,
callbacks=[early_stop, PrintDot()])

plot_history(history)
```

يوضح الرسم البياني أنهفي مجموعة التحقق، يكون متوسط الخطأ عادةً حوالي 2 +/- ميلافي الغالون. هل هذا جيد؟ سنترك هذا القرار لك.

دعونا نرى كيف يتم تعميم النموذج باستخدام مجموعة الاختبار، والتي لم نستخدمها عند تدريب النموذج. يوضح هذا إلى أي مدى يُتوقع من هذا النموذج أن يتنبأ عندما نستخدمهفي العالم الحقيقي:

```
loss, mae, mse = model.evaluate(normed_test_data, test_labels,
verbose=0)
print("Testing set Mean Abs Error: {:5.2f} MPG".format(mae))
```

```
Testing set Mean Abs Error:  1.97 MPG
```

الآن، دعنا نضع تنبؤات على النموذج للتنبؤ بكفاءة الوقود:

```
test_predictions = model.predict(normed_test_data).flatten()
```

```
plt.scatter(test_labels, test_predictions)
plt.xlabel('True Values [MPG]')
plt.ylabel('Predictions [MPG]')
plt.axis('equal')
plt.axis('square')
plt.xlim([0,plt.xlim()[1]])
plt.ylim([0,plt.ylim()[1]])
_ = plt.plot([-100, 100], [-100, 100])
```

يبدو أن النموذج تنبأ بشكل جيد. آمل أن تكون هذه المقالة قد أعجبتك، للتنبؤ بكفاءة استهلاك الوقود باستخدام التعلم الآلي.

11) تصنيف النص باستخدام TensorFlow في التعلم الآلي Text Classification with TensorFlow in Machine Learning

في هذه المقالة، سأقدم لك نموذج تصنيف النص باستخدام TensorFlow في مراجعات الأفلام على أنها إيجابية أو سلبية باستخدام نص المراجعات. هذه مشكلة تصنيف ثنائي، وهي نوع مهم وقابل للتطبيق على نطاق واسع من مشاكل التعلم الآلي.

تصنيف النص باستخدام TensorFlow

سأوجهك عبر التطبيق الأساسي لنقل التعلم باستخدام TensorFlow Hub وKeras. سأستخدم مجموعة بيانات IMDB التي تحتوي على نصوص 50،000 فيلم من قاعدة بيانات الأفلام على الإنترنت. وهي مقسمة إلى 25000 تقييم للتدريب و25000 تقييم للاختبار. يتم موازنة مجموعات التدريب والاختبار بطريقة تحتوي على عدد متساوٍ من المراجعات الإيجابية والسلبية.

الآن، لنبدأ بهذه المهمة الخاصة بتصنيف النص باستخدام TensorFlow عن طريق استيراد بعض المكتبات الضرورية:

```
import numpy as np

import tensorflow as tf

!pip install tensorflow-hub
!pip install tensorflow-datasets
import tensorflow_hub as hub
import tensorflow_datasets as tfds

print("Version: ", tf.__version__)
print("Eager mode: ", tf.executing_eagerly())
print("Hub version: ", hub.__version__)
print("GPU is", "available" if
tf.config.experimental.list_physical_devices("GPU") else "NOT
AVAILABLE")
```

على الرغم من أن مجموعة البيانات التي أستخدمها هنا متاحة للتنزيل عبر الإنترنت، إلا أنني سأقوم ببساطة بتحميل البيانات باستخدام TensorFlow. هذا يعني أنك لست بحاجة إلى تنزيل مجموعة البيانات من أي مصادر خارجية. الآن، سأقوم ببساطة بتحميل البيانات وتقسيمها إلى مجموعات تدريب واختبار:

```
# Split the training set into 60% and 40%, so we'll end up with 15,000
examples
# for training, 10,000 examples for validation and 25,000 examples for
testing.
train_data, validation_data, test_data = tfds.load(
    name="imdb_reviews",
    split=('train[:60%]', 'train[60%:]', 'test'),
```

```
    as_supervised=True)
```

استكشاف البيانات

دعونا نلقي نظرة على البيانات لمعرفة ما سنعمل معه. سأقوم ببساطة بطباعة أول 10 عينات من مجموعة البيانات:

```
train_examples_batch, train_labels_batch =
next(iter(train_data.batch(10)))
```

```
train_examples_batch
```

الآن، دعنا نطبع أول 10 تسميات من مجموعة البيانات:

```
train_labels_batch
```

Output: <tf.Tensor: shape=(10,), dtype=int64, numpy=array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0])>

بناء نموذج تصنيف النص

لبناء نموذج لمهمة تصنيف النص باستخدام TensorFlow، سأستخدم نموذجًا تم تدريبه مسبقًا مقدمًا من TensorFlow والذي يُعرف باسم TensorFlow Hub. دعنا أولاً ننشئ طبقة Keras تستخدم نموذج TensorFlow Hub في الجمل المضمنة، ونجربهافي بعض نماذج الإدخال:

```
embedding = "https://tfhub.dev/google/tf2-preview/gnews-swivel-
20dim/1"
hub_layer = hub.KerasLayer(embedding, input_shape=[],
                           dtype=tf.string, trainable=True)
hub_layer(train_examples_batch[:3])
```

الآن قم ببناء النموذج على مجموعة البيانات الكاملة:

```
model = tf.keras.Sequential()

model.add(hub_layer)

model.add(tf.keras.layers.Dense(16, activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 20)                400020
_________________________________________________________________
dense (Dense)                (None, 16)                336
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 400,373
Trainable params: 400,373
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج

الآن، سأقوم بتجميع compile النموذج باستخدام دالة الخسارة ومحسن آدم:

```
model.compile(optimizer='adam',

loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

تدريب نموذج تصنيف النص

تدريب النموذج لمدة 20 حقبةفي مجموعات صغيرة من 512 عينة. هذه 20 تكرارًا على جميع عينات الموتر x_train و y_train. أثناء التدريب، راقب خطأ النموذج ودقته على 10000 عينة في مجموعة التحقق من الصحة:

```
history = model.fit(train_data.shuffle(10000).batch(512),
                    epochs=20,
                    validation_data=validation_data.batch(512),
                    verbose=1)
```

تقييم النموذج

ودعونا نرى كيف يعمل نموذج تصنيف النص. سيتم إرجاع قيمتين. معدل الخسارة والدقة:

```
results = model.evaluate(test_data.batch(512), verbose=2)

for name, value in zip(model.metrics_names, results):
  print("%s: %.3f" % (name, value))
```

```
49/49 - 3s - loss: 0.3217 - accuracy: 0.8553
loss: 0.322
accuracy: 0.855
```

لذلك حقق نموذج تصنيف النص لدينا معدل دقة بنسبة 85في المائة وهو موضع تقدير بشكل عام. أتمنى أن تكون قد أحببت هذا المقال حول نموذج تصنيف النص باستخدام TensorFlow.

12) تصنيف الصور باستخدام TensorFlow في التعلم الآلي Image Classification with TensorFlow in Machine Learning

في هذه المقالة، سأشرح كيف يمكننا تدريب نموذج الشبكة العصبية لمهمة تصنيف الصور باستخدام TensorFlow. بالنسبة لأولئك الجدد على TensorFlow، فإن TensorFlow عبارة عن نظام أساسي مفتوح المصدر وشامل للتعلم الآلي. لديها نظام بيئي شامل ومرن من الأدوات والمكتبات وموارد المجتمع التي تسمح للباحثين بدفع التطورات المتطورةفي تعلم الآلة، والمطورين لبناء ونشر التطبيقات القائمة على التعلم الآلي بسهولة.

ما هو تصنيف الصور؟

تصنيف الصورة هو عملية تصنيف وتسمية مجموعات البكسل أو المتجهاتفي صورة ما وفقًا لقواعد محددة. يمكن تصميم قانون التصنيف باستخدام خاصية واحدة أو أكثر من الخصائص الطيفية spectral أو التركيبية textural.

تصنيف الصور باستخدام TensorFlow

الآن، يمكن أيضًا إجراء تصنيف الصور باستخدام نماذج أقل تعقيدًا مقدمة من Scikit-Learn، فلماذا TensorFlow. باستخدام TensorFlow، يمكننا بناء شبكة عصبية لمهمة تصنيف الصور. من خلال بناء شبكة عصبية يمكننا اكتشاف أنماط خفية أكثر من مجرد تصنيف. لنبدأ الآن بمهمة تصنيف الصور باستخدام TensorFlow عن طريق استيراد بعض الحزم الضرورية:

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
```

استيراد مجموعة بيانات Fashion MNIST

تم تصميم Fashion MNIST كبديل سريع لمجموعة بيانات MNIST الكلاسيكية ـ غالبًا ما تستخدم كـ "Hello, World" لبرامج التعلم الآلي للرؤية الحاسوبية. تحتوي مجموعة بيانات MNIST على صور لأرقام مكتوبة بخط اليد (0، 1، 2، إلخ) بتنسيق مماثل لصور الملابس التي سأستخدمها لمهمة تصنيف الصور باستخدام TensorFlow.

```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()
```

في مجموعة البيانات، يتم تعيين كل صورة في تسمية واحدة. نظرًا لعدم تحديد أسماء الفئات في مجموعة البيانات، نحتاج إلى تخزينها هنا حتى نتمكن من استخدامها لاحقًا عند عرض الصور:

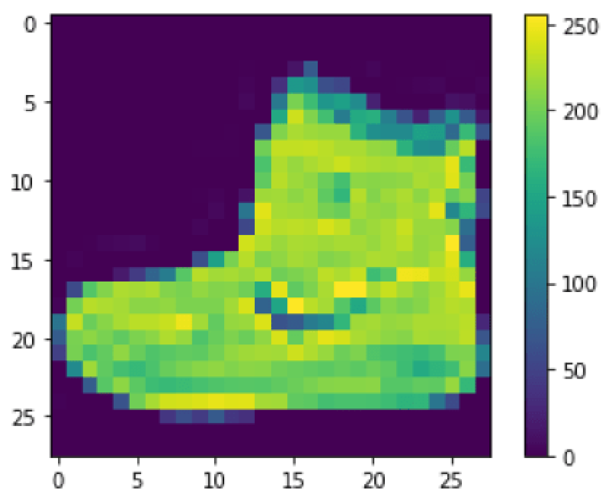
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

معالجة البيانات

بالنسبة لمهمة تصنيف الصور هذه باستخدام TensorFlow، يجب معالجة البيانات مسبقًا قبل تدريب الشبكة العصبية. إذا قمت بفحص الإطار الأول لمجموعة التدريب، فستجد أن قيم البكسل بين 0 و255:



```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



سأقوم الآن بتوسيع نطاق هذه القيم إلى نطاق من 0 إلى 1 قبل تمريرها إلى نموذج الشبكة العصبية. للقيام بذلك، نحتاج إلى قسمة القيم على 255. يجب معالجة مجموعة التدريب ومجموعة الاختبار بالطريقة نفسها:

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

للتحقق من أن البيانات بالتنسيق الصحيح وللتحقق من استعدادنا لإنشاء وتدريب الشبكة العصبية لتصنيف الصور باستخدام TensorFlow، فلنعرض أول 25 صورة لمجموعة التدريب ونعرض اسم الفئة تحت كل صورة:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
```



تصنيف الصور باستخدام TensorFlow: بناء نموذج

الآن لبناء الشبكة العصبية لمهمة تصنيف الصور باستخدام TensorFlow، نحتاج أولاً إلى تكوين طبقات النموذج ثم المضي قدمًافي تجميع compiling النموذج.

إعداد الطبقات

اللبنة الأساسية للشبكات العصبية هي طبقاتها. تعمل الطبقات عن طريق استخراج التمثيلات من البيانات التي يتم إدخالها فيها. معظم التعلم العميق، النماذج تتضمن عمل طبقات بسيطة معًا. الآن، دعنا ننشئ طبقات شبكتنا العصبية:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

تجميع النموذج

الآن، دعنا نمضي قدمًا في تجميع compiling نموذجنا:

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

تصنيف الصور باستخدام TensorFlow: تدريب النموذج

الآن، دعنا ندرب الشبكة العصبية على مهمة تصنيف الصور باستخدام TensorFlow، وقم بعمل تنبؤات بشأنها:

```
#Fitting the Model
model.fit(train_images, train_labels, epochs=10)
#Evaluating Accuracy
test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)
print('\nTest accuracy:', test_acc)
```

```
Test accuracy: 0.8817999958992004
```

```
#Make Predictions
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
predictions[0]
```

```
array([1.1349098e-09, 1.0395625e-09, 3.4154518e-10, 8.3033120e-12,
    6.5739442e-10, 5.9645530e-03, 9.4151291e-09, 1.1747092e-02,
    8.7000714e-08, 9.8228824e-01], dtype=float32)
```

التوقع هو مصفوفة من 10 أرقام. إنها تمثل "ثقة confidence" النموذجفي أن الصورة تتطابق مع كل قطعة من الملابس العشرة المختلفة. دعنا نرى التصنيف الذي يحتوي على أعلى قيمة ثقة:

```
np.argmax(predictions[0])
```

Output: 9

وبالتالي، فإن النموذج مقتنع تمامًا بأن هذه الصورة عبارة عن حذاء للكاحل ankle boot أو class_names [9]. يوضح فحص تسمية الاختبار أن هذا التصنيف صحيح:

```
test_labels[0]
```

9

الآن، سأقوم بإنشاء دالة مساعدة لرسم توقعاتنا:

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

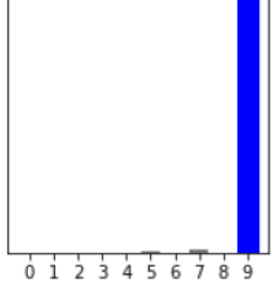
  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

التحقق من التنبؤات

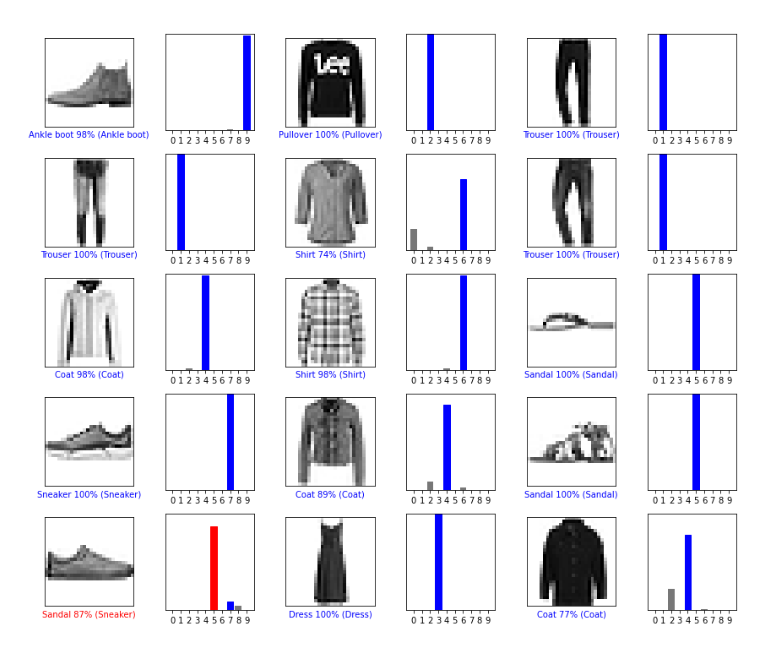
دعونا نلقي نظرة على الإطار 0 من التوقعات وجدول التنبؤ. تسميات التنبؤ الصحيحة باللون الأزرق وعلامات التوقع غير الصحيحة باللون الأحمر:

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```


Ankle boot 98% (Ankle boot)




```
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```



يبدو الإخراج رائعًا، فقط الأحذية التي يتم التعرف عليها على أنها خاطئة مثل الصنادل. آمل أن تكون قد أحببت هذه المقالة حول تصنيف الصور باستخدام التعلم الآلي.

13) التعرف على الصور باستخدام التعلم الآلي باستخدام PyTorch
Image Recognition with Machine Learning using PyTorch

لا يبذل البشر أي جهد للتمييز بين كلب أو قطة أو صحن طائر. لكن هذه العملية يصعب على الكمبيوتر محاكاتها: تبدو سهلة فقط لأن الله يصمم أدمغتنا بشكل جيد للغاية للتعرف على الصور. أحد الأمثلة الشائعة على التعرف على الصور باستخدام التعلم الآلي هو التعرف الضوئي على الأحرف optical character recognition.في هذه المقالة، سوف آخذك خلال بناء نموذج التعرف على الصور باستخدام التعلم الآلي باستخدام PyTorch.

ما هو PyTorch؟

قبل الغوص في هذه المهمة، دعنا نفهم أولاً ما هو PyTorch. PyTorch هي مكتبة لبرامج بايثون التي تجعل من السهل إنشاء نماذج التعلم العميق. مثل بايثون للبرمجة، توفر PyTorch مقدمة رائعة للتعلم العميق.في الوقت نفسه، أثبتت PyTorch أنها مؤهلة تمامًا للاستخدام في السياقات المهنية للعمل عالي المستوى في العالم الحقيقي.

التعرف على الصور باستخدام التعلم الآلي

بالنسبة لمهمة التعرف على الصور،في هذه المقالة، سأستخدم حزمة TorchVision التي تحتوي على بعض من أفضل بنى الشبكات العصبية أداءً للرؤية الحاسوبية، مثل AlexNet. كما يوفر وصولاً سهلاً إلى مجموعات البيانات مثل ImageNet والأدوات المساعدة الأخرى للتعرف على تطبيقات الرؤية الحاسوبية في PyTorch.

يمكن العثور على النماذج المحددة مسبقًا في torchvision.models.

```
from torchvision import models
dir(models)
```

```
['AlexNet',
 'DenseNet',
 'GoogLeNet',
 'GoogLeNetOutputs',
 'Inception3',
 'InceptionOutputs',
 'MNASNet',
 'MobileNetV2',
 'ResNet',
 'ShuffleNetV2',
 'SqueezeNet',
 'VGG',
 '_GoogLeNetOutputs',
 '_InceptionOutputs',
```

```
'__builtins__',
'__cached__',
'__doc__',
'__file__',
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'_utils',
'alexnet',
'densenet',
'densenet121',
'densenet161',
'densenet169',
'densenet201',
'detection',
'googlenet',
'inception',
'inception_v3',
'mnasnet',
'mnasnet0_5',
'mnasnet0_75',
'mnasnet1_0',
'mnasnet1_3',
'mobilenet',
'mobilenet_v2',
'quantization',
'resnet',
'resnet101',
'resnet152',
'resnet18',
'resnet34',
'resnet50',
'resnext101_32x8d',
'resnext50_32x4d',
'segmentation',
'shufflenet_v2_x0_5',
'shufflenet_v2_x1_0',
'shufflenet_v2_x1_5',
'shufflenet_v2_x2_0',
'shufflenetv2',
'squeezenet',
'squeezenet1_0',
'squeezenet1_1',
'utils',
'vgg',
'vgg11',
'vgg11_bn',
```

'vgg13',
'vgg13_bn',
'vgg16',
'vgg16_bn',
'vgg19',
'vgg19_bn',
'video',
'wide_resnet101_2',
'wide_resnet50_2']

تشير الأسماء الكبيرة إلى فئات بايثون التي تنفذ العديد من النماذج الشائعة. الأسماء الصغيرة هي دوال يدوية تُرجع أنماطًا تم إنشاء مثيل لها من هذه الفئات، أحيانًا بمجموعات مختلفة من المعلمات.

AlexNet

لتشغيل بُنية AlexNet على صورة إدخال، يمكننا إنشاء مثيل لفئة AlexNet. هنا كيفية القيام بذلك:

```
alexnet = models.AlexNet()
```

في هذه المرحلة، alexnet هو كائن يدير بُنية AlexNet. ليس من الضروري بالنسبة لنا أن نفهم تفاصيل هذه البُنيةفي هذا الوقت.في الوقت الحالي، يعد AlexNet مجرد كائن معتم يمكن تسميته كدالة.

من خلال تزويد alexnet ببيانات الإدخال ذات الحجم الدقيق، سنقوم بإجراء نقل مباشر عبر الشبكة. بمعنى آخر، ستمر المدخلات من خلال المجموعة الأولى من الخلايا العصبية، والتي سيتم نقل مخرجاتها إلى المجموعة التالية من الخلايا العصبية، حتى الإخراج النهائي.

ResNet

باستخدام طريقة resnet101، يمكننا الآن إنشاء مثيل لشبكة عصبية تلافيفية مكونة من 101 طبقة. لنقم الآن بإنشاء مثيل للشبكة. سنقوم بتمرير وسيطة ستطلب من الدالة تنزيل أوزان resnet101 التي تم تكوينها على مجموعة بيانات ImageNet، مع 1.2 مليون صورة و1000 فئة:

```
resnet = models.resnet101(pretrained=True)
resnet
```

الآن، يمكن استدعاء متغير resnet كدالة. قبل أن نتمكن من القيام بذلك، ومع ذلك، نحتاج إلى المعالجة المسبقة لصور الإدخال بحيث تكون بالحجم الصحيح وتكون قيمها (الألوان)في نفس النطاق الرقمي تقريبًا. للقيام بذلك، نحتاج إلى استخدام وحدة torchvision التي توفر التحولات، والتي ستتيح لنا تحديد خطوط أنابيب دوال المعالجة الأولية الأساسية بسرعة:

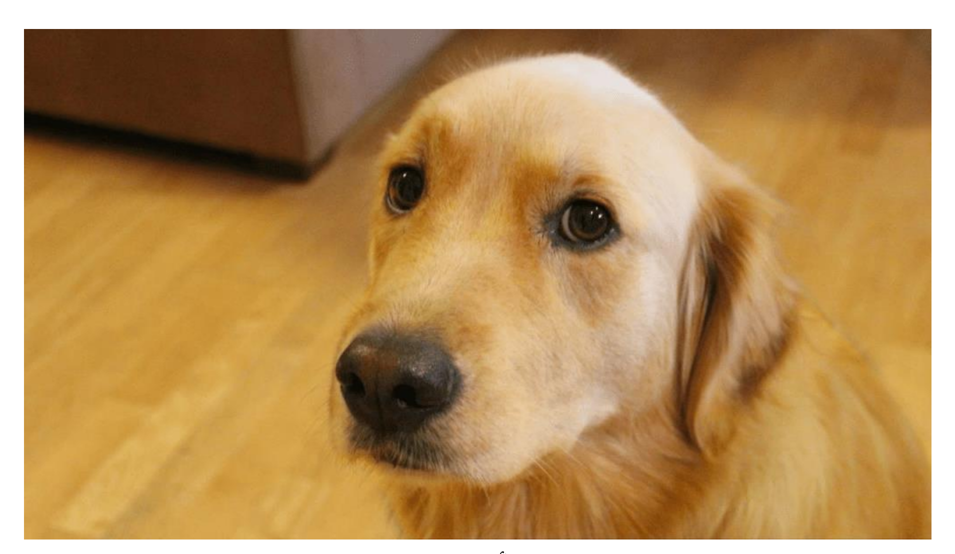
```
from torchvision import transforms
preprocess = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(
            mean=[0.485, 0.456, 0.406],
            std=[0.229, 0.224, 0.225]
        )])
```

في هذه الحالة، حددنا دالة المعالجة المسبقة التي ستعمل على قياس صورة الإدخال إلى 256 × 256، واقتصاص الصورة إلى 224 × 224 حول المركز، وتحويلها إلى موتر، وتسوية مكونات RGBالخاصة بها. (أحمر، أخضر، أزرق) بحيث يكون لديهم وسائط محددة وانحرافات معيارية.

التعرف على الصور

الآن يمكننا استخدام صورة لمهمة التعرف على الصور باستخدام نموذجنا. لقد التقطت صورة لكلب. يمكننا البدء بتحميل صورة من نظام الملفات المحلي باستخدام Pillow، وهي وحدة معالجة الصور لبايثون:

```
from google.colab import files
uploaded = files.upload()
from PIL import Image
img = Image.open("dog.png")
img
```



بعد ذلك، نحتاج إلى تمرير الصورة عبر خط أنابيب المعالجة المسبقة لدينا للتعرف على الصور:

```
img_t = preprocess(img)
```

يمكننا الآن إعادة تشكيل موتر الإدخال واقتصاصه وتسويته بالطريقة التي تتوقعها الشبكة:

```
import torch
batch_t = torch.unsqueeze(img_t, 0)
resnet.eval()
out = resnet(batch_t)
out
```

تشغيل نموذج التعرف على الصور

تسمى عملية تشغيل نموذج مدرب على بيانات جديدة الاستدلال inferenceفي دوائر التعلم العميق. من أجل عمل استنتاجات لنموذج التعرف على الصور هذا، نحتاج إلى وضع الشبكةفي وضع التقييم. فلننقم الآن بتحميل الملف الذي يحتوي على 1000 تصنيف لفئات مجموعة البياناتImageNet :

```
with open('imagenet_classes.txt') as f:
    labels = [line.strip() for line in f.readlines()]
_, index = torch.max(out, 1)

percentage = torch.nn.functional.softmax(out, dim=1)[0] * 100
labels[index[0]], percentage[index[0]].item()
```

```
('golden retriever', 96.29334259033203)
```

هذا يعطينا شيئًا يشبه تقريبًا الثقة التي يمتلكها النموذجفي تنبؤاته.في هذه الحالة، فإن النموذج متأكد بنسبة 96٪ أنه يعرف أن ما ينظر إليه هو كلب جولدن ريتريفر.

آمل أن تكون قد أحببت هذه المقالة حول التعرف على الصور باستخدام التعلم الآلي باستخدام PyTorch.

14) نظام توصية للأزياء Fashion Recommendation System

في هذه المقالة، سوف أطلعك على كيفية إنشاء نظام توصية للأزياء باستخدام التعلم الآلي الذي سيعمل مثل توصيات التسوق عبر الإنترنت المخصصة للغاية. ولكن قبل المضي قدمًا، عليك أن تعرف ما هو نظام التوصية recommendation system.

نظام التوصية هو نظام مبرمج للتنبؤ بالعناصر المفضلةفي المستقبل من مجموعة كبيرة من المجموعات. يعمل نظام التوصية إما باستخدام تفضيلات المستخدم أو باستخدام العناصر الأكثر تفضيلاً من قبل جميع المستخدمين. التحدي الرئيسيفي بناء نظام توصية للأزياء هو أنها صناعة ديناميكية للغاية. يتغير كثيرًا عندما يتعلق الأمر بالمواسم والمهرجانات والظروف الوبائية مثل فيروس كورونا وغيرها الكثير.

نظام توصية للأزياء مع تعلم الآلة

على عكس المجالات الأخرى، لا ينبغي أن تستند توصيات الأزياء فقط على الذوق الشخصي والنشاط السابق للعميل. هناك العديد من العوامل الخارجية (العديد منها عاطفية) التي تجعل إنشاء نظام توصية للأزياء أكثر تعقيداً. يجب أن تؤخذ التصورات العامةفي الاعتبار، وكذلك قواعد الموضة وقواعد اللباس والتوجهات الحالية.

دعنا نتعمق الآنفي بناء نظام توصية للأزياء باستخدام التعلم الآلي. سأبدأ ببساطة باستيراد جميع الحزم التي نحتاجها لهذه المهمة:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import gdown
from fastai.vision import *
from fastai.metrics import accuracy, top_k_accuracy
from annoy import AnnoyIndex
import zipfile
import time
from google.colab import drive
%matplotlib inline
```

الآن، تحتاج إلى جمع البياناتفي محرك google الخاص بك، وعليك لصق عناوين URL لهذه الروابط لاستيراد مجموعة البياناتفي دفتر ملاحظاتك:

```
# get the meta data
url = 'https://drive.google.com/uc?id=0B7EVK8r0v71pWnFiNlNGTVloLUk'
output = 'list_category_cloth.txt'
gdown.download(url, output, quiet=False)

url = 'https://drive.google.com/uc?id=0B7EVK8r0v71pTGNoWkhZeVpzbFk'
output = 'list_category_img.txt'
```

```
gdown.download(url, output, quiet=False)

url = 'https://drive.google.com/uc?id=0B7EVK8r0v71pdS1FMlNreEwtc1E'
output = 'list_eval_partition.txt'
gdown.download(url, output, quiet=False)
```

الآن، دعنا نحصل على جميع الصور من محرك google الخاص بنا:

```
# get the images
root_path = './'
url =
'https://drive.google.com/uc?id=1j5fCPgh0gnY6v7ChkWlgnnHH6unxuAbb'
output = 'img.zip'
gdown.download(url, output, quiet=False)
with zipfile.ZipFile("img.zip","r") as zip_ref:
    zip_ref.extractall(root_path)
```

سأقوم الآن ببعض خطوات تحضير البيانات وتنظيف البيانات لتأطير البيانات بطريقة مفيدة:

```
category_list = []
image_path_list = []
data_type_list = []
# category names
with open('list_category_cloth.txt', 'r') as f:
    for i, line in enumerate(f.readlines()):
        if i &gt; 1:
            category_list.append(line.split(' ')[0])

# category map
with open('list_category_img.txt', 'r') as f:
    for i, line in enumerate(f.readlines()):
        if i &gt; 1:
            image_path_list.append([word.strip() for word in
line.split(' ') if len(word) &gt; 0])


# train, valid, test
with open('list_eval_partition.txt', 'r') as f:
    for i, line in enumerate(f.readlines()):
        if i &gt; 1:
            data_type_list.append([word.strip() for word in
line.split(' ') if len(word) &gt; 0])

data_df = pd.DataFrame(image_path_list, columns=['image_path',
'category_number'])
data_df['category_number'] = data_df['category_number'].astype(int)
data_df = data_df.merge(pd.DataFrame(data_type_list,
columns=['image_path', 'dataset_type']), on='image_path')
data_df['category'] = data_df['category_number'].apply(lambda x:
category_list[int(x) - 1])
data_df = data_df.drop('category_number', axis=1)
```

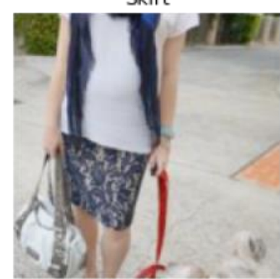
الآن، سيقوم الكود أدناه بتحويل جميع الصور إلى التضمينات embeddings:

```
train_image_list = ImageList.from_df(df=data_df, path=root_path,
cols='image_path').split_by_idxs(
    (data_df[data_df['dataset_type']=='train'].index),

(data_df[data_df['dataset_type']=='val'].index)).label_from_df(cols='c
ategory')
test_image_list = ImageList.from_df(df=data_df[data_df['dataset_type']
== 'test'], path=root_path, cols='image_path')

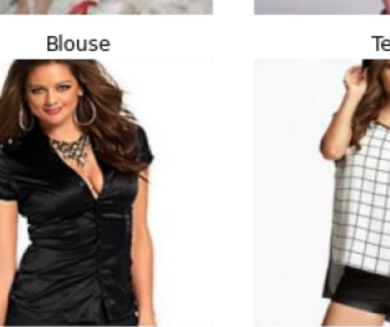
data = train_image_list.transform(get_transforms(),
size=224).databunch(bs=128).normalize(imagenet_stats)
data.add_test(test_image_list)
data.show_batch(rows=3, figsize=(8,8))
```

Shorts Dress Skirt

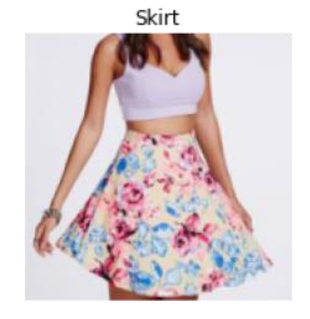












الآن، سأقوم ببعض الخطوات لنقل التعلم باستخدام مكتبة resnet و PyTorch للحصول على معدل التعلم:

```
# see models available: https://docs.fast.ai/vision.models.html
# many options for Resnet, the numbers are the number of layers.
```

```
# More layers are generally more accurate but take longer to train:
resnet18, resnet34, resnet50, resnet101, resnet152
# get top 1 and top 5 accuracy
def train_model(data, pretrained_model, model_metrics):
    learner = cnn_learner(data, pretrained_model,
metrics=model_metrics)
    learner.model = torch.nn.DataParallel(learner.model)
    learner.lr_find()
    learner.recorder.plot(suggestion=True)
    return learner

pretrained_model = models.resnet18 # simple model that can be trained
on free tier
# pretrained_model = models.resnet50 # need pro tier, model I used

model_metrics = [accuracy, partial(top_k_accuracy, k=1),
partial(top_k_accuracy, k=5)]
learner = train_model(data, pretrained_model, model_metrics)
learner.fit_one_cycle(10, max_lr=1e-02)
```

الآن، دعنا نقيم نموذج نقل التعلم:

```
interp = ClassificationInterpretation.from_learner(learner)
interp.plot_top_losses(9,        largest=False,        figsize=(15,11),
heatmap_thresh=5)
```

Prediction/Actual/Loss/Probability















يبدو الناتج جيداً، الآن قبل المضي قدمًا، دعنا نحفظ هذا النموذج حتى نتمكن من استخدامه بسهولة لمهامنا المستقبلية:

```
# saving the model (temporary, will lose model once environment
resets)
learner.save('resnet-fashion')
```

الآن، سأستخدم طريقة FastAI لاسترداد صور التضمينات:

```
class SaveFeatures():
    features=None
    def __init__(self, m):
        self.hook = m.register_forward_hook(self.hook_fn)
        self.features = None
    def hook_fn(self, module, input, output):
        out = output.detach().cpu().numpy()
        if isinstance(self.features, type(None)):
            self.features = out
        else:
            self.features = np.row_stack((self.features, out))
    def remove(self):
        self.hook.remove()

  # load the trained model
def load_learner(data, pretrained_model, model_metrics, model_path):
    learner = cnn_learner(data, pretrained_model,
metrics=model_metrics)
    learner.model = torch.nn.DataParallel(learner.model)
    learner = learner.load(model_path)
    return learner

pretrained_model = models.resnet18 # simple model that can be trained
on free tier
# pretrained_model = models.resnet50 # need pro tier

model_metrics = [accuracy, partial(top_k_accuracy, k=1),
partial(top_k_accuracy, k=5)]
# if gdrive not mounted:
drive.mount('/content/gdrive')


model_path = "/content/gdrive/My Drive/resnet18-fashion"
# model_path = "/content/gdrive/My Drive/resnet50-fashion"
learner = load_learner(data, pretrained_model, model_metrics,
model_path)
```

أتمنى أن تكون قد فهمت شيئًا من العملية المذكورة أعلاه، والآن سأستخدم طريقة أقرب جيران nearest neighbours لإنشاء نظام توصية للأزياء:

```
# takes time to populate the embeddings for each image
# Get 2nd last layer of the model that stores the embedding for the
image representations
# the last linear layer is the output layer.
```

```
saved_features = SaveFeatures(learner.model.module[1][4])
_= learner.get_preds(data.train_ds)
_= learner.get_preds(DatasetType.Valid)
```

أخيرًا، نقوم بإدراج عمليات التضمين الخاصة بـ 12 عنصرًا (أو أكثر) من تحديد المستخدم في قائمة ومتوسط قيم التضمينات في كل من الأبعاد؛ يؤدي هذا إلى إنشاء كائن شبح ghost object يمثل القيمة الإجمالية لجميع العناصر المحددة.

يمكننا بعد ذلك العثور على أقرب جار لهذا الكائن الشبح:

```
# prepare the data for generating recommendations (exlcude test data)
# get the embeddings from trained model
img_path = [str(x) for x in (list(data.train_ds.items)
+list(data.valid_ds.items))]
label = [data.classes[x] for x in (list(data.train_ds.y.items)
+list(data.valid_ds.y.items))]
label_id = [x for x in (list(data.train_ds.y.items)
+list(data.valid_ds.y.items))]
data_df_ouput = pd.DataFrame({'img_path': img_path, 'label': label,
'label_id': label_id})
data_df_ouput['embeddings'] =
np.array(saved_features.features).tolist()
# Using Spotify's Annoy
def get_similar_images_annoy(annoy_tree, img_index,
number_of_items=12):
    start = time.time()
    img_id, img_label  = data_df_ouput.iloc[img_index, [0, 1]]
    similar_img_ids = annoy_tree.get_nns_by_item(img_index,
number_of_items+1)
    end = time.time()
    print(f'{(end - start) * 1000} ms')
    # ignore first item as it is always target image
    return img_id, img_label, data_df_ouput.iloc[similar_img_ids[1:]]


# for images similar to centroid
def get_similar_images_annoy_centroid(annoy_tree, vector_value,
number_of_items=12):
    start = time.time()
    similar_img_ids = annoy_tree.get_nns_by_vector(vector_value,
number_of_items+1)
    end = time.time()
    print(f'{(end - start) * 1000} ms')
    # ignore first item as it is always target image
    return data_df_ouput.iloc[similar_img_ids[1:]]


def show_similar_images(similar_images_df, fig_size=[10,10],
hide_labels=True):
    if hide_labels:
        category_list = []
        for i in range(len(similar_images_df)):
```

```
            # replace category with blank so it wont show in display

category_list.append(CategoryList(similar_images_df['label_id'].values
*0,
                                              [''] *
len(similar_images_df)).get(i))
    else:
        category_list = [learner.data.train_ds.y.reconstruct(y) for y
in similar_images_df['label_id']]
    return learner.data.show_xys([open_image(img_id) for img_id in
similar_images_df['img_path']],
                                category_list, figsize=fig_size)
  # more tree = better approximation
ntree = 100
#"angular", "euclidean", "manhattan", "hamming", or "dot"
metric_choice = 'angular'

annoy_tree = AnnoyIndex(len(data_df_ouput['embeddings'][0]),
metric=metric_choice)

# # takes a while to build the tree
for i, vector in enumerate(data_df_ouput['embeddings']):
    annoy_tree.add_item(i, vector)
_  = annoy_tree.build(ntree)
```

اختبار نظام توصية للأزياء

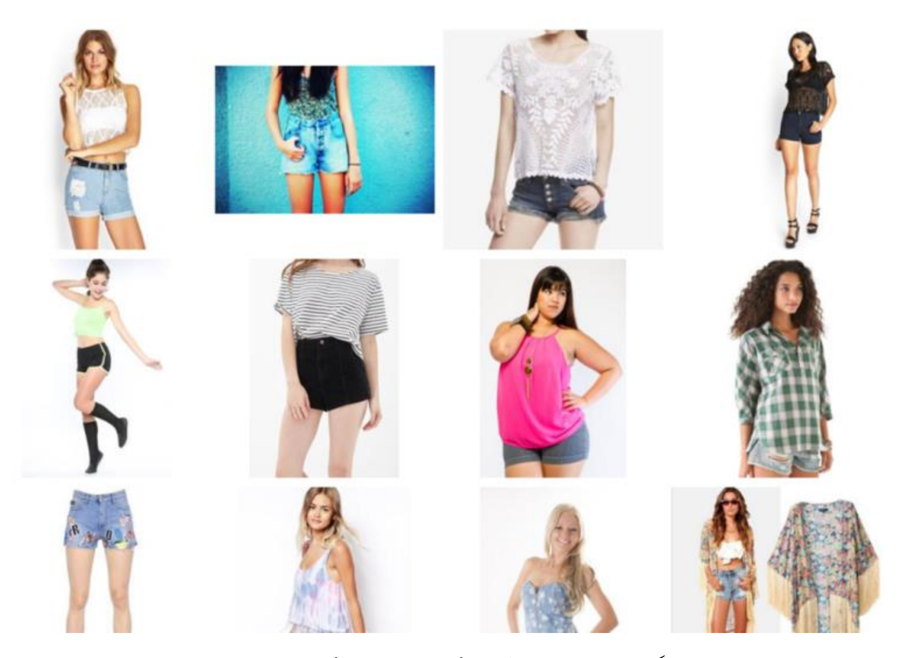
الآن، دعنا نختبر نظام توصيات الأزياء لدينا. لهذا، نحتاج إلى إنشاء بعض الرموز. أولاً، دعنا نرى التوصيات لأي شيء يتعلق بالسراويل القصيرة "shorts":

```
def centroid_embedding(outfit_embedding_list):
    number_of_outfits = outfit_embedding_list.shape[0]
    length_of_embedding = outfit_embedding_list.shape[1]
    centroid = []
    for i in range(length_of_embedding):
        centroid.append(np.sum(outfit_embedding_list[:,
i])/number_of_outfits)
    return centroid
 # shorts
outfit_img_ids = [109938, 106385, 113703, 98666, 113467, 120667,
20840, 8450, 142843, 238607, 124505,222671]
outfit_embedding_list = []
for img_index in outfit_img_ids:
    outfit_embedding_list.append(data_df_ouput.iloc[img_index, 3])

outfit_embedding_list = np.array(outfit_embedding_list)
outfit_centroid_embedding = centroid_embedding(outfit_embedding_list)
outfits_selected = data_df_ouput.iloc[outfit_img_ids]

similar_images_df    =    get_similar_images_annoy_centroid(annoy_tree,
outfit_centroid_embedding, 30)
```

وأخيرًا، سنرى توصيات "السراويل القصيرة":



هذه استجابة جيدة جداً من نموذجنا. آمل أن تكون قد أحببت هذه المقالة حول نظام توصية الموضة مع التعلم الآلي.

15) التعرف على الكيان المسمى (NER) Named Entity Recognition

الكيان المسمى Named Entity يعني أي شيء يمثل كائنًا حقيقيًا مثل شخص أو مكان أو أي منظمة أو أي منتج له اسم. على سبيل المثال – "اسمي أمان وأنا مدرب تعلم الآلة".في هذه الجملة، يُطلق على اسم "أمان" والحقل أو الموضوع "التعلم الآلي" والمهنة "المدرب" كيانات.

في التعرف على الكيانات المسماة (NER) في التعلم الآلي، يعد التعرف على الكيانات المسماة (NER) مهمة معالجة اللغة الطبيعية لتحديد الكيانات المسماة في جزء معين من النص.

هل سبق لك استخدام برنامج يعرف باسم Grammarly؟ يحدد جميع التهجئات وعلامات الترقيم غير الصحيحة في النص ويصححها. لكنها لا تفعل شيئًا مع الكيانات المسماة، لأنها تستخدم نفس التقنية أيضًا.في هذه المقالة، سوف آخذك خلال مهمة التعرف على الكيانات المسماة (NER) باستخدام التعلم الآلي.

تحميل البيانات الخاصة بالتعرف على الكيان المحدد (NER)

يمكن تنزيل مجموعة البيانات التي سأستخدمها لهذه المهمة بسهولة من هنا. الآن أول شيء سأفعله هو تحميل البيانات وإلقاء نظرة عليها لمعرفة ما أعمل به. لذلك دعونا نستورد مكتبة pandas وتحميل البيانات:

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
data = pd.read_csv('ner_dataset.csv', encoding= 'unicode_escape')
data.head()
```

	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	NaN	of	IN	O
2	NaN	demonstrators	NNS	O
3	NaN	have	VBP	O
4	NaN	marched	VBN	O

في البيانات، يمكننا أن نرى أن الكلمات مقسمة إلى أعمدة تمثل ميزتنا X، وسيمثل عمود العلامة في اليمين التسمية Y.

تحضير البيانات للشبكات العصبية

سأقوم بتدريب شبكة عصبية على مهمة التعرف على الكيان المحدد (NER). لذلك نحن بحاجة إلى إجراء بعض التعديلات على البيانات لإعدادها بهذه الطريقة بحيث يمكن أن تتناسب بسهولة مع شبكة محايدة. سأبدأ هذه الخطوة باستخراج التعيينات المطلوبة لتدريب الشبكة العصبية:

```
from itertools import chain
def get_dict_map(data, token_or_tag):
    tok2idx = {}
    idx2tok = {}

    if token_or_tag == 'token':
        vocab = list(set(data['Word'].to_list()))
    else:
        vocab = list(set(data['Tag'].to_list()))

    idx2tok = {idx:tok for  idx, tok in enumerate(vocab)}
    tok2idx = {tok:idx for  idx, tok in enumerate(vocab)}
    return tok2idx, idx2tok
token2idx, idx2token = get_dict_map(data, 'token')
tag2idx, idx2tag = get_dict_map(data, 'tag')
```

سأقوم الآن بتحويل الأعمدةفي البيانات لاستخراج البيانات المتسلسلة لشبكتنا العصبية:

```
data['Word_idx'] = data['Word'].map(token2idx)
data['Tag_idx'] = data['Tag'].map(tag2idx)
data_fillna = data.fillna(method='ffill', axis=0)
# Groupby and collect columns
data_group = data_fillna.groupby(
['Sentence #'],as_index=False
)['Word', 'POS', 'Tag', 'Word_idx', 'Tag_idx'].agg(lambda x: list(x))
```

الآن سأقسم البيانات إلى مجموعات تدريب واختبار. سأقوم بإنشاء دالة لتقسيم البيانات لأن طبقات LSTM تقبل تسلسلات من نفس الطول فقط. لذلك يجب أن تكون كل جملة تظهر على هيئة عدد صحيحفي البيانات مبطن بنفس الطول:

```
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

def get_pad_train_test_val(data_group, data):

    #get max token and tag length
    n_token = len(list(set(data['Word'].to_list())))
    n_tag = len(list(set(data['Tag'].to_list())))

    #Pad tokens (X var)
```

```
    tokens = data_group['Word_idx'].tolist()
    maxlen = max([len(s) for s in tokens])
    pad_tokens = pad_sequences(tokens, maxlen=maxlen, dtype='int32',
padding='post', value= n_token - 1)

    #Pad Tags (y var) and convert it into one hot encoding
    tags = data_group['Tag_idx'].tolist()
    pad_tags = pad_sequences(tags, maxlen=maxlen, dtype='int32',
padding='post', value= tag2idx["O"])
    n_tags = len(tag2idx)
    pad_tags = [to_categorical(i, num_classes=n_tags) for i in
pad_tags]

    #Split train, test and validation set
    tokens_, test_tokens, tags_, test_tags =
train_test_split(pad_tokens, pad_tags, test_size=0.1, train_size=0.9,
random_state=2020)
    train_tokens, val_tokens, train_tags, val_tags =
train_test_split(tokens_,tags_,test_size = 0.25,train_size =0.75,
random_state=2020)

    print(
        'train_tokens length:', len(train_tokens),
        '\ntrain_tokens length:', len(train_tokens),
        '\ntest_tokens length:', len(test_tokens),
        '\ntest_tags:', len(test_tags),
        '\nval_tokens:', len(val_tokens),
        '\nval_tags:', len(val_tags),
    )

    return train_tokens, val_tokens, test_tokens, train_tags,
val_tags, test_tags

train_tokens, val_tokens, test_tokens, train_tags, val_tags, test_tags
= get_pad_train_test_val(data_group, data)
```

```
train_tokens length: 32372
train_tokens length: 32372
test_tokens length: 4796
test_tags: 4796
val_tokens: 10791 val_tags: 10791
```

تدريب الشبكة العصبية للتعرف على الكيانات المسماة (NER)

الآن، سأشرع في تدريب بُنية الشبكة العصبية لنموذجنا. فلنبدأ باستيراد جميع الحزم التي نحتاجها لتدريب شبكتنا العصبية:

```
import numpy as np
import tensorflow
from tensorflow.keras import Sequential, Model, Input
from tensorflow.keras.layers import LSTM, Embedding, Dense,
TimeDistributed, Dropout, Bidirectional
from tensorflow.keras.utils import plot_model
```

```
from numpy.random import seed
seed(1)
tensorflow.random.set_seed(2)
```

ستأخذ الطبقة أدناه الأبعاد من طبقة LSTM وستعطي الحد الأقصى للطول والحد الأقصى للعلامات كإخراج:

```
input_dim = len(list(set(data['Word'].to_list())))+1
output_dim = 64
input_length = max([len(s) for s in data_group['Word_idx'].tolist()])
n_tags = len(tag2idx)
```

الآن سوف أقوم بإنشاء دالة مساعدة والتي ستساعدنافي إعطاء ملخص لكل طبقة من نموذج الشبكة العصبية للتعرف على الكيانات المسماة (NER):

```
def get_bilstm_lstm_model():
    model = Sequential()

    # Add Embedding layer
    model.add(Embedding(input_dim=input_dim, output_dim=output_dim,
input_length=input_length))

    # Add bidirectional LSTM
    model.add(Bidirectional(LSTM(units=output_dim,
return_sequences=True, dropout=0.2, recurrent_dropout=0.2), merge_mode
= 'concat'))

    # Add LSTM
    model.add(LSTM(units=output_dim, return_sequences=True,
dropout=0.5, recurrent_dropout=0.5))

    # Add timeDistributed Layer
    model.add(TimeDistributed(Dense(n_tags, activation="relu")))

    #Optimiser
    # adam = k.optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999)

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

سأقوم الآن بإنشاء دالة مساعدة لتدريب نموذج التعرف على الكيان المحدد:

```
def train_model(X, y, model):
    loss = list()
    for i in range(25):
        # fit model for one epoch on this sequence
        hist = model.fit(X, y, batch_size=1000, verbose=1, epochs=1,
validation_split=0.2)
```

```
        loss.append(hist.history['loss'][0])
    return loss
```

الدرايفر كود:

```
results = pd.DataFrame()
model_bilstm_lstm = get_bilstm_lstm_model()
plot_model(model_bilstm_lstm)
results['with_add_lstm']          =          train_model(train_tokens,
np.array(train_tags), model_bilstm_lstm)
```

سيعطي النموذج الناتج النهائي بعد تشغيله لمدة 25 حقبة. لذلك سوف يستغرق الأمر بعض الوقت للتشغيل.

اختبار نموذج التعرف على الكيان المسماة (NER):

الآن دعنا نختبر نموذجنا على جزء من النص:

```
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')
text = nlp('Hi, My name is Aman Kharwal \n I am from India \n I want
to work with Google \n Steve Jobs is My Inspiration')
displacy.render(text, style = 'ent', jupyter=True)
```

Hi, My name is Aman Kharwal **PERSON**

I am from India **GPE**

I want to work with Google **ORG**

Steve Jobs **PERSON** is My Inspiration

يمكننا أن نرى نتيجة جيدة جداً من نموذجنا. آمل أن تكون قد أحببت هذه المقالة حول التعرف على الكيانات المسماة (NER) باستخدام التعلم الآلي.

16) نموذج الترجمة الآلية Machine Translation Model

تعد الترجمة الآلية Machine Translation واحدة من أكثر المهام تحديًافي الذكاء الاصطناعي والتي تعمل من خلال التحقق من استخدام البرنامج لترجمة نص أو خطاب من لغة إلى أخرى. في هذه المقالة، سوف آخذك عبر الترجمة الآلية باستخدام الشبكات العصبية.

في نهاية هذه المقالة، ستتعلم تطوير نموذج ترجمة آلية باستخدام الشبكات العصبية وبايثون. سأستخدم اللغة الإنجليزية كمدخل وسنقوم بتدريب نموذج الترجمة الآلية لدينا لإعطاء المخرجات باللغة الفرنسية. لنبدأ الآن باستيراد جميع المكتبات التي نحتاجها لهذه المهمة:

```
import collections
import helper
import numpy as np
import project_tests as tests
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model
from keras.layers import GRU, Input, Dense, TimeDistributed,
Activation, RepeatVector, Bidirectional
from keras.layers.embeddings import Embedding
from keras.optimizers import Adam
from keras.losses import sparse_categorical_crossentropy
```

سأقوم أولاً بإنشاء دالتين لتحميل البيانات ودالة أخرى لاختبار بياناتنا:

```
import os
def load_data(path):
    """
    Load dataset
    """
    input_file = os.path.join(path)
    with open(input_file, "r") as f:
        data = f.read()

    return data.split('\n')
def _test_model(model, input_shape, output_sequence_length,
french_vocab_size):
    if isinstance(model, Sequential):
        model = model.model

    assert model.input_shape == (None, *input_shape[1:]),\
        'Wrong input shape. Found input shape {} using parameter
input_shape={}'.format(model.input_shape, input_shape)

    assert model.output_shape == (None, output_sequence_length,
french_vocab_size),\
        'Wrong output shape. Found output shape {} using parameters
output_sequence_length={} and french_vocab_size={}'\
            .format(model.output_shape, output_sequence_length,
french_vocab_size)
```

```
    assert len(model.loss_functions) &amp;gt; 0,\
        'No loss function set.  Apply the `compile` function to the
model.'

    assert sparse_categorical_crossentropy in model.loss_functions,\
        'Not using `sparse_categorical_crossentropy` function for loss.
```

لنقم الآن بتحميل البيانات وإلقاء نظرة على بعض الأفكار من البيانات، حيث تحتوي مجموعة البيانات التي أستخدمها هنا على عبارة باللغة الإنجليزية مع ترجمتها:

```
english_sentences = helper.load_data('data/small_vocab_en')
french_sentences = helper.load_data('data/small_vocab_fr')
print('Dataset Loaded')
```

تم تحميل مجموعة البيانات

```
for sample_i in range(2):
    print('small_vocab_en Line {}:  {}'.format(sample_i + 1,
english_sentences[sample_i]))
    print('small_vocab_fr   Line   {}:      {}'.format(sample_i   +   1,
french_sentences[sample_i]))
```

```
small_vocab_en Line 1: new jersey is sometimes quiet during autumn , and it is snowy in april .
small_vocab_fr Line 1: new jersey est parfois calme pendant l' automne , et il est neigeux en avril .
small_vocab_en Line 2: the united states is usually chilly during july , and it is usually freezing in
november . small_vocab_fr Line 2: les états-unis est généralement froid en juillet , et il gèle
habituellement en novembre .
```

أثناء قيامنا بترجمة اللغة، سيتم تحديد مدى تعقيد هذه المشكلة من خلال تعقيد المفردات. كلما كانت مفردات لغتنا أكثر تعقيداً، كلما كانت مشكلتنا أكثر تعقيداً. دعونا نلقي نظرة على البيانات لمعرفة البيانات المعقدة التي نتعامل معها:

```
english_words_counter = collections.Counter([word for sentence in
english_sentences for word in sentence.split()])
french_words_counter = collections.Counter([word for sentence in
french_sentences for word in sentence.split()])
print('{} English words.'.format(len([word for sentence in
english_sentences for word in sentence.split()])))
print('{} unique English words.'.format(len(english_words_counter)))
print('10 Most common words in the English dataset:')
print('"' + '"
"'.join(list(zip(*english_words_counter.most_common(10)))[0]) + '"')
print()
print('{} French words.'.format(len([word for sentence in
french_sentences for word in sentence.split()])))
print('{} unique French words.'.format(len(french_words_counter)))
print('10 Most common words in the French dataset:')
print('"'                              +                              '"
"'.join(list(zip(*french_words_counter.most_common(10)))[0]) + '"')
```

```
1823250 English words.
227 unique English words.
```

```
10 Most common words in the English dataset:
"is" "," "." "in" "it" "during" "the" "but" "and" "sometimes"

1961295 French words.
355 unique French words.
10 Most common words in the French dataset:
"est" "." "," "en" "il" "les" "mais" "et" "la" "parfois"
```

المعالجة المسبقة للبيانات

في التعلم الآلي أينما نتعامل مع أي نوع من القيم النصية، نحتاج أولاً إلى تحويل القيم النصية إلى تسلسلات من الأعداد الصحيحة باستخدام طريقتين أساسيتين مثل الترميز Tokenize والحشو Padding. لنبدأ الآن بالترميز Tokenization:

```python
def tokenize(x):
    x_tk = Tokenizer(char_level = False)
    x_tk.fit_on_texts(x)
    return x_tk.texts_to_sequences(x), x_tk
text_sentences = [
    'The quick brown fox jumps over the lazy dog .',
    'By Jove , my quick study of lexicography won a prize .',
    'This is a short sentence .']
text_tokenized, text_tokenizer = tokenize(text_sentences)
print(text_tokenizer.word_index)
print()
for sample_i, (sent, token_sent) in enumerate(zip(text_sentences,
text_tokenized)):
    print('Sequence {} in x'.format(sample_i + 1))
    print('  Input:  {}'.format(sent))
    print('  Output: {}'.format(token_sent))
```

```
{'the': 1, 'quick': 2, 'a': 3, 'brown': 4, 'fox': 5, 'jumps': 6, 'over': 7, 'lazy': 8, 'dog': 9, 'by': 10, 'jove': 11,
'my': 12, 'study': 13, 'of': 14, 'lexicography': 15, 'won': 16, 'prize': 17, 'this': 18, 'is': 19, 'short': 20,
'sentence': 21}

Sequence 1 in x
  Input:  The quick brown fox jumps over the lazy dog .
  Output: [1, 2, 4, 5, 6, 7, 1, 8, 9]
Sequence 2 in x
  Input:  By Jove , my quick study of lexicography won a prize .
  Output: [10, 11, 12, 2, 13, 14, 15, 16, 3, 17]
Sequence 3 in x
  Input:  This is a short sentence .
  Output: [18, 19, 3, 20, 21]
```

دعنا الآن نستخدم طريقة الحشو لعمل كل التسلسلات بنفس الطول:

```python
def pad(x, length=None):
    if length is None:
        length = max([len(sentence) for sentence in x])
    return pad_sequences(x, maxlen = length, padding = 'post')
tests.test_pad(pad)
# Pad Tokenized output
test_pad = pad(text_tokenized)
for sample_i, (token_sent, pad_sent) in enumerate(zip(text_tokenized,
test_pad)):
    print('Sequence {} in x'.format(sample_i + 1))
```

```
    print('  Input:  {}'.format(np.array(token_sent)))
    print('  Output: {}'.format(pad_sent))
```

```
Sequence 1 in x
 Input:  [1 2 4 5 6 7 1 8 9]
 Output: [1 2 4 5 6 7 1 8 9 0]
Sequence 2 in x
 Input:  [10 11 12  2 13 14 15 16  3 17]
 Output: [10 11 12  2 13 14 15 16  3 17]
Sequence 3 in x
 Input:  [18 19  3 20 21]
 Output: [18 19  3 20 21  0  0  0  0  0]
```

المعالجة المسبقة لخط أنابيب للترجمة الآلية

دعنا الآن نحدد دالة المعالجة المسبقة لإنشاء خط أنابيب لمهمة الترجمة الآلية حتى نتمكن من استخدام هذا النموذج في المستقبل أيضًا:

```
def preprocess(x, y):
    preprocess_x, x_tk = tokenize(x)
    preprocess_y, y_tk = tokenize(y)
preprocess_x = pad(preprocess_x)
    preprocess_y = pad(preprocess_y)
# Keras's sparse_categorical_crossentropy function requires the labels
to be in 3 dimensions
    preprocess_y = preprocess_y.reshape(*preprocess_y.shape, 1)
return preprocess_x, preprocess_y, x_tk, y_tk
preproc_english_sentences, preproc_french_sentences,
english_tokenizer, french_tokenizer =\
    preprocess(english_sentences, french_sentences)

max_english_sequence_length = preproc_english_sentences.shape[1]
max_french_sequence_length = preproc_french_sentences.shape[1]
english_vocab_size = len(english_tokenizer.word_index)
french_vocab_size = len(french_tokenizer.word_index)
print('Data Preprocessed')
print("Max English sentence length:", max_english_sequence_length)
print("Max French sentence length:", max_french_sequence_length)
print("English vocabulary size:", english_vocab_size)
print("French vocabulary size:", french_vocab_size)
```

```
Data Preprocessed
Max English sentence length: 15
Max French sentence length: 21
English vocabulary size: 199
French vocabulary size: 344
```

تدريب شبكة عصبية للترجمة الآلية

الآن، سأقوم هنا بتدريب نموذج باستخدام الشبكات العصبية. لنبدأ بإنشاء دالة مساعدة:

```
def logits_to_text(logits, tokenizer):
    index_to_words = {id: word for word, id in
tokenizer.word_index.items()}
    index_to_words[0] = '&amp;lt;PAD&amp;gt;'
return ' '.join([index_to_words[prediction] for prediction in
np.argmax(logits, 1)])
print('`logits_to_text` function loaded.')
```

`logits_to_text` function loaded.

الآن سأقوم بتدريب نموذج RNN والذي سيكون بمثابة قاعدة جيدة جداً لتسلسلاتنا التي يمكنها ترجمة اللغة الإنجليزية إلى الفرنسية:

```
def simple_model(input_shape, output_sequence_length,
english_vocab_size, french_vocab_size):
    learning_rate = 1e-3
    input_seq = Input(input_shape[1:])
    rnn = GRU(64, return_sequences = True)(input_seq)
    logits = TimeDistributed(Dense(french_vocab_size))(rnn)
    model = Model(input_seq, Activation('softmax')(logits))
    model.compile(loss = sparse_categorical_crossentropy,
                 optimizer = Adam(learning_rate),
                 metrics = ['accuracy'])

    return model
tests.test_simple_model(simple_model)
tmp_x = pad(preproc_english_sentences, max_french_sequence_length)
tmp_x = tmp_x.reshape((-1, preproc_french_sentences.shape[-2], 1))
# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024,
epochs=10, validation_split=0.2)
# Print prediction(s)
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0],
french_tokenizer))
```

```
Train on 110288 samples, validate on 27573 samples
Epoch 1/10
110288/110288 [==============================] - 9s 82us/step - loss: 3.5162 - acc: 0.4027 - val_loss:
nan - val_acc: 0.4516
Epoch 2/10
110288/110288 [==============================] - 7s 64us/step - loss: 2.4823 - acc: 0.4655 - val_loss:
nan - val_acc: 0.4838
Epoch 3/10
110288/110288 [==============================] - 7s 63us/step - loss: 2.2427 - acc: 0.5016 - val_loss:
nan - val_acc: 0.5082
Epoch 4/10
110288/110288 [==============================] - 7s 64us/step - loss: 2.0188 - acc: 0.5230 - val_loss:
nan - val_acc: 0.5428
Epoch 5/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.8418 - acc: 0.5542 - val_loss:
nan - val_acc: 0.5685
Epoch 6/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.7258 - acc: 0.5731 - val_loss:
nan - val_acc: 0.5811
Epoch 7/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.6478 - acc: 0.5871 - val_loss:
```

```
Epoch 8/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.5850 - acc: 0.5940 - val_loss:
nan - val_acc: 0.5977
Epoch 9/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.5320 - acc: 0.5996 - val_loss:
nan - val_acc: 0.6027
Epoch 10/10
110288/110288 [==============================] - 7s 64us/step - loss: 1.4874 - acc: 0.6037 - val_loss:
nan - val_acc: 0.6039
new jersey est parfois parfois en en et il est est en en <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD>
```

أعطانا نموذج RNN دقة 60في المائة فقط، دعنا نستخدم شبكة عصبية أكثر تعقيداً لتدريب نموذجنا بدقة أفضل. سأقوم الآن بتدريب نموذجنا باستخدام RNN مع التضمين embedding. يمثل التضمين متجهًا لكلمة قريبة جداً من كلمة مشابهة في العالم ذي البعد n. يمثل n هنا حجم متجهات التضمين:

```
from keras.models import Sequential
def embed_model(input_shape, output_sequence_length,
english_vocab_size, french_vocab_size):
    learning_rate = 1e-3
    rnn = GRU(64, return_sequences=True, activation="tanh")

    embedding = Embedding(french_vocab_size, 64,
input_length=input_shape[1])
    logits = TimeDistributed(Dense(french_vocab_size,
activation="softmax"))

    model = Sequential()
    #em can only be used in first layer --&amp;gt; Keras Documentation
    model.add(embedding)
    model.add(rnn)
    model.add(logits)
    model.compile(loss=sparse_categorical_crossentropy,
                  optimizer=Adam(learning_rate),
                  metrics=['accuracy'])

    return model
tests.test_embed_model(embed_model)
tmp_x = pad(preproc_english_sentences, max_french_sequence_length)
tmp_x = tmp_x.reshape((-1, preproc_french_sentences.shape[-2]))
embeded_model = embed_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
    french_vocab_size)
embeded_model.fit(tmp_x, preproc_french_sentences, batch_size=1024,
epochs=10, validation_split=0.2)
print(logits_to_text(embeded_model.predict(tmp_x[:1])[0],
french_tokenizer))
```

```
Train on 110288 samples, validate on 27573 samples
Epoch 1/10
110288/110288 [==============================] - 8s 68us/step - loss: 3.7877 - acc: 0.4018 - val_loss:
nan - val_acc: 0.4093
Epoch 2/10
110288/110288 [==============================] - 7s 65us/step - loss: 2.7258 - acc: 0.4382 - val_loss:
nan - val_acc: 0.5152
Epoch 3/10
110288/110288 [==============================] - 7s 65us/step - loss: 2.0359 - acc: 0.5453 - val_loss:
nan - val_acc: 0.6068
Epoch 4/10
110288/110288 [==============================] - 7s 65us/step - loss: 1.4586 - acc: 0.6558 - val_loss:
nan - val_acc: 0.6967
Epoch 5/10
110288/110288 [==============================] - 7s 65us/step - loss: 1.1346 - acc: 0.7308 - val_loss:
nan - val_acc: 0.7561
Epoch 6/10
110288/110288 [==============================] - 7s 65us/step - loss: 0.9358 - acc: 0.7681 - val_loss:
nan - val_acc: 0.7825
Epoch 7/10
110288/110288 [==============================] - 7s 65us/step - loss: 0.8057 - acc: 0.7917 - val_loss:
nan - val_acc: 0.7993
Epoch 8/10
110288/110288 [==============================] - 7s 65us/step - loss: 0.7132 - acc: 0.8095 - val_loss:
nan - val_acc: 0.8173
Epoch 9/10
110288/110288 [==============================] - 7s 65us/step - loss: 0.6453 - acc: 0.8229 - val_loss:
nan - val_acc: 0.8313
Epoch 10/10
110288/110288 [==============================] - 7s 64us/step - loss: 0.5893 - acc: 0.8355 - val_loss:
nan - val_acc: 0.8401
new jersey est parfois calme au l'automne et il il est neigeux en en <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD>
```

لذلك، أدى نموذج RNN الخاص بنا مع التضمين إلى دقة جيدة جداً تبلغ 84في المائة. آمل أن تكون قد أحببت هذا المقال عن الترجمة الآلية باستخدام الشبكات العصبية وبايثون.

17) كشف معالم الوجه Face Landmarks Detection

هل فكرت يومًافي كيفية تمكن Snapchat من تطبيق فلاتر مذهلة وفقًا لوجهك؟ تمت برمجته لاكتشاف بعض العلامات على وجهك لإبراز فلتر وفقًا لتلك العلامات.في التعلم الآلي، تُعرف هذه العلامات باسم معالم الوجه Face Landmarks.في هذه المقالة، سأوجهك إلى كيفية اكتشاف معالم الوجوه باستخدام التعلم الآلي.

الآن، سأبدأ ببساطة باستيراد جميع المكتبات التي نحتاجها لهذه المهمة. سأستخدم PyTorch في هذه المقالة لمواجهة اكتشاف المعالم باستخدام التعلم العميق. لنستورد جميع المكتبات:

```
import time
import cv2
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import imutils
import matplotlib.image as mpimg
from collections import OrderedDict
from skimage import io, transform
from math import *
import xml.etree.ElementTree as ET

import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision.transforms.functional as TF
from torchvision import datasets, models, transforms
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
```

تنزيل مجموعة بيانات DLIB

مجموعة البيانات التي سأختارها هنا لاكتشاف معالم الوجهفي مجموعة بيانات DLIB الرسمية التي تتكون من أكثر من 6666 صورة بأبعاد مختلفة. سيقوم الكود أدناه بتنزيل مجموعة البيانات وفك الضغط لمزيد من الاستكشاف:

```
%%capture
if not
os.path.exists('/content/ibug_300W_large_face_landmark_dataset'):
    !wget
http://dlib.net/files/data/ibug_300W_large_face_landmark_dataset.tar.g
z
    !tar -xvzf 'ibug_300W_large_face_landmark_dataset.tar.gz'
    !rm -r 'ibug_300W_large_face_landmark_dataset.tar.gz'
```

التمثيل البياني لمجموعة البيانات

الآن، دعنا نلقي نظرة على ما نعمل معه، لنرى جميع تنظيف البيانات وفرص المعالجة المسبقة التي نحتاج إلى خوضها. فيما يلي مثال على صورة من مجموعة البيانات التي أخذناها لهذه المهمة.

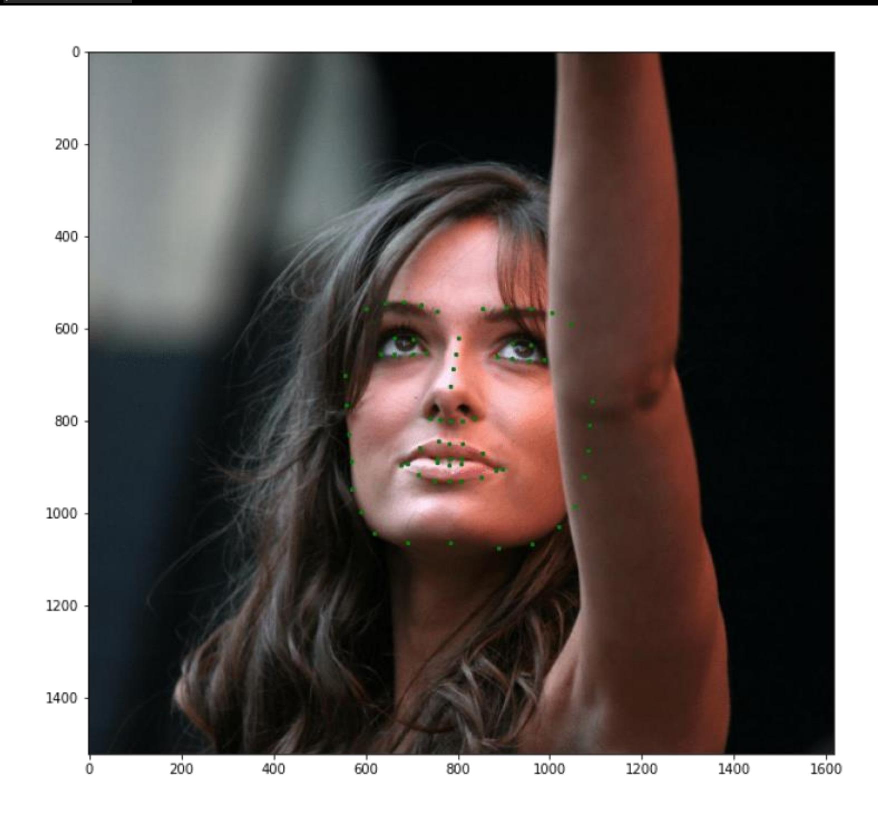
```
file =
open('ibug_300W_large_face_landmark_dataset/helen/trainset/100032540_1
.pts')
points = file.readlines()[3:-1]

landmarks = []

for point in points:
    x,y = point.split(' ')
    landmarks.append([floor(float(x)), floor(float(y[:-1]))])

landmarks = np.array(landmarks)

plt.figure(figsize=(10,10))
plt.imshow(mpimg.imread('ibug_300W_large_face_landmark_dataset/helen/t
rainset/100032540_1.jpg'))
plt.scatter(landmarks[:,0], landmarks[:,1], s = 5, c = 'g')
plt.show()
```



يمكنك أن ترى أن الوجه يغطي مساحة أقل بكثيرفي الصورة. إذا كنا سنستخدم هذه الصورةفي الشبكة العصبية، فستأخذ الخلفية أيضًا. لذا، مثلما نقوم بإعداد بيانات نصية، سنقوم بإعداد مجموعة بيانات الصور هذه لمزيد من الاستكشاف.

تكوين فئات مجموعة البيانات

دعنا الآن نتعمق أكثرفي الفئات والتسمياتفي مجموعة البيانات. يتكون labels_ibug_300W_train.xml من الصور المدخلة والمعالم والمربع المحيط لاقتصاص الوجه. سوف أقوم بتخزين كل هذه القيمفي القائمة حتى نتمكن من الوصول إليها بسهولة أثناء عملية التدريب.

```
class Transforms():
    def __init__(self):
        pass

    def rotate(self, image, landmarks, angle):
        angle = random.uniform(-angle, +angle)

        transformation_matrix = torch.tensor([
            [+cos(radians(angle)), -sin(radians(angle))],
            [+sin(radians(angle)), +cos(radians(angle))]
        ])

        image = imutils.rotate(np.array(image), angle)

        landmarks = landmarks - 0.5
        new_landmarks = np.matmul(landmarks, transformation_matrix)
        new_landmarks = new_landmarks + 0.5
        return Image.fromarray(image), new_landmarks

    def resize(self, image, landmarks, img_size):
        image = TF.resize(image, img_size)
        return image, landmarks

    def color_jitter(self, image, landmarks):
        color_jitter = transforms.ColorJitter(brightness=0.3,
                                              contrast=0.3,
                                              saturation=0.3,
                                              hue=0.1)
        image = color_jitter(image)
        return image, landmarks

    def crop_face(self, image, landmarks, crops):
        left = int(crops['left'])
        top = int(crops['top'])
        width = int(crops['width'])
        height = int(crops['height'])

        image = TF.crop(image, top, left, height, width)
```

```
        img_shape = np.array(image).shape
        landmarks = torch.tensor(landmarks) - torch.tensor([[left,
top]])
        landmarks = landmarks / torch.tensor([img_shape[1],
img_shape[0]])
        return image, landmarks

    def __call__(self, image, landmarks, crops):
        image = Image.fromarray(image)
        image, landmarks = self.crop_face(image, landmarks, crops)
        image, landmarks = self.resize(image, landmarks, (224, 224))
        image, landmarks = self.color_jitter(image, landmarks)
        image, landmarks = self.rotate(image, landmarks, angle=10)

        image = TF.to_tensor(image)
        image = TF.normalize(image, [0.5], [0.5])
        return image, landmarks

class FaceLandmarksDataset(Dataset):

    def __init__(self, transform=None):

        tree =
ET.parse('ibug_300W_large_face_landmark_dataset/labels_ibug_300W_train
.xml')
        root = tree.getroot()

        self.image_filenames = []
        self.landmarks = []
        self.crops = []
        self.transform = transform
        self.root_dir = 'ibug_300W_large_face_landmark_dataset'

        for filename in root[2]:
            self.image_filenames.append(os.path.join(self.root_dir,
filename.attrib['file']))

            self.crops.append(filename[0].attrib)

            landmark = []
            for num in range(68):
                x_coordinate = int(filename[0][num].attrib['x'])
                y_coordinate = int(filename[0][num].attrib['y'])
                landmark.append([x_coordinate, y_coordinate])
            self.landmarks.append(landmark)

        self.landmarks = np.array(self.landmarks).astype('float32')

        assert len(self.image_filenames) == len(self.landmarks)

    def __len__(self):
        return len(self.image_filenames)

    def __getitem__(self, index):
        image = cv2.imread(self.image_filenames[index], 0)
```

```
        landmarks = self.landmarks[index]

        if self.transform:
            image, landmarks = self.transform(image, landmarks,
self.crops[index])

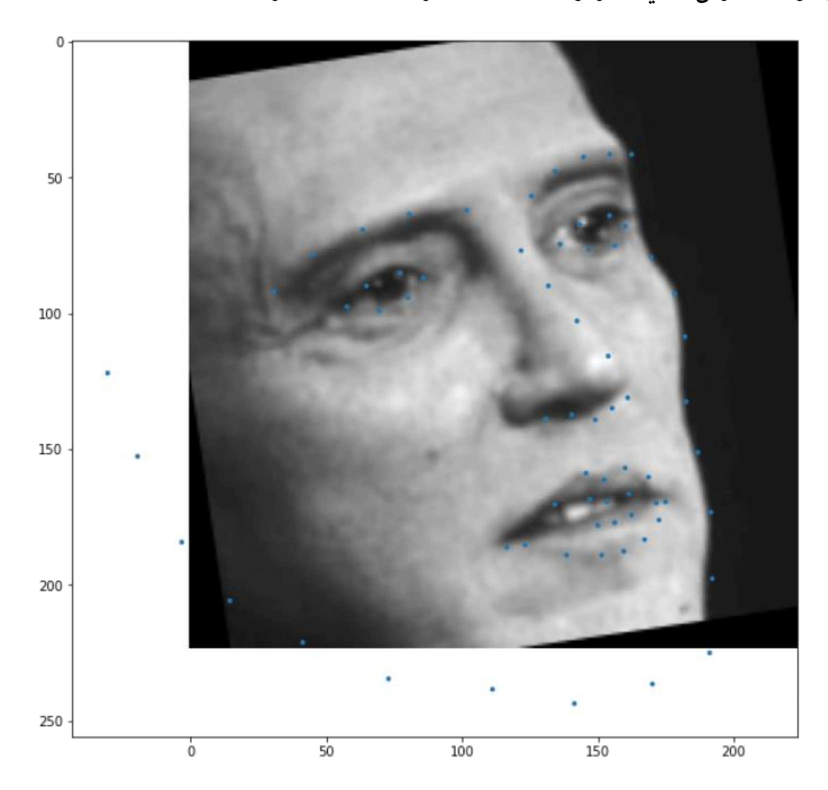
        landmarks = landmarks - 0.5

        return image, landmarks

dataset = FaceLandmarksDataset(Transforms())
```

رسم تحويلات التدريب:

الآن دعونا نلقي نظرة سريعة على ما قمنا به حتى الآن. سوف نرسم مجموعة البيانات فقط من خلال إجراء التحويل الذي ستوفره الفئات المذكورة أعلاه لمجموعة البيانات:



تقسيم مجموعة البيانات للتدريب والتنبؤ بمعالم الوجه

الآن، للمضي قدمًا، سأقسم مجموعة البيانات إلى مجموعة تدريب ومجموعة بيانات التحقق من الصحة:

```
# split the dataset into validation and test sets
len_valid_set = int(0.1*len(dataset))
len_train_set = len(dataset) - len_valid_set
```

```
print("The length of Train set is {}".format(len_train_set))
print("The length of Valid set is {}".format(len_valid_set))

train_dataset , valid_dataset,  =
torch.utils.data.random_split(dataset , [len_train_set,
len_valid_set])

# shuffle and batch the datasets
train_loader = torch.utils.data.DataLoader(train_dataset,
batch_size=64, shuffle=True, num_workers=4)
valid_loader = torch.utils.data.DataLoader(valid_dataset, batch_size=8,
shuffle=True, num_workers=4)
```

```
The length of Train set is 6000
The length of Valid set is 666
```

اختبار شكل بيانات الإدخال:

```
images, landmarks = next(iter(train_loader))

print(images.shape)
print(landmarks.shape)
```

```
torch.Size([64, 1, 224, 224])
torch.Size([64, 68, 2])
```

تعريف نموذج اكتشاف معالم الوجه

الآن سأستخدم ResNet18 كإطار عمل أساسي لدينا. سوف أقوم بتعديل الطبقتين الأولى والأخيرة بحيث تتناسب الطبقات بسهولة مع غرضنا:

```
class Network(nn.Module):
    def __init__(self,num_classes=136):
        super().__init__()
        self.model_name='resnet18'
        self.model=models.resnet18()
        self.model.conv1=nn.Conv2d(1, 64, kernel_size=7, stride=2,
padding=3, bias=False)
        self.model.fc=nn.Linear(self.model.fc.in_features,
num_classes)

    def forward(self, x):
        x=self.model(x)
        return x
```

دوال المساعدة:

```
import sys

def print_overwrite(step, total_step, loss, operation):
    sys.stdout.write('\r')
    if operation == 'train':
        sys.stdout.write("Train Steps: %d/%d  Loss: %.4f " % (step,
total_step, loss))
    else:
```

```
        sys.stdout.write("Valid Steps: %d/%d  Loss: %.4f " % (step,
total_step, loss))

    sys.stdout.flush()
```

تدريب الشبكة العصبية لاكتشاف معالم الوجه

سأستخدم الآن الخطأ التربيعي المتوسط بين معالم الوجه الحقيقية والمتوقعة:

```
torch.autograd.set_detect_anomaly(True)
network = Network()
network.cuda()

criterion = nn.MSELoss()
optimizer = optim.Adam(network.parameters(), lr=0.0001)

loss_min = np.inf
num_epochs = 10

start_time = time.time()
for epoch in range(1,num_epochs+1):

    loss_train = 0
    loss_valid = 0
    running_loss = 0

    network.train()
    for step in range(1,len(train_loader)+1):

        images, landmarks = next(iter(train_loader))

        images = images.cuda()
        landmarks = landmarks.view(landmarks.size(0),-1).cuda()

        predictions = network(images)

        # clear all the gradients before calculating them
        optimizer.zero_grad()

        # find the loss for the current step
        loss_train_step = criterion(predictions, landmarks)

        # calculate the gradients
        loss_train_step.backward()

        # update the parameters
        optimizer.step()

        loss_train += loss_train_step.item()
        running_loss = loss_train/step

        print_overwrite(step, len(train_loader), running_loss,
'train')

    network.eval()
```

```
    with torch.no_grad():

        for step in range(1,len(valid_loader)+1):

            images, landmarks = next(iter(valid_loader))

            images = images.cuda()
            landmarks = landmarks.view(landmarks.size(0),-1).cuda()

            predictions = network(images)

            # find the loss for the current step
            loss_valid_step = criterion(predictions, landmarks)

            loss_valid += loss_valid_step.item()
            running_loss = loss_valid/step

            print_overwrite(step, len(valid_loader), running_loss,
'valid')

    loss_train /= len(train_loader)
    loss_valid /= len(valid_loader)

    print('\n--------------------------------------------------')
    print('Epoch: {}  Train Loss: {:.4f}  Valid Loss:
{:.4f}'.format(epoch, loss_train, loss_valid))
    print('--------------------------------------------------')

    if loss_valid < loss_min:
        loss_min = loss_valid
        torch.save(network.state_dict(),
'/content/face_landmarks.pth')
        print("\nMinimum Validation Loss of {:.4f} at epoch
{}/{}".format(loss_min, epoch, num_epochs))
        print('Model Saved\n')
print('Training Complete')
print("Total Elapsed Time : {} s".format(time.time()-start_time))
```

توقع معالم الوجه

دعنا الآن نستخدم النموذج الذي دربناه أعلاه على الصور التي لم يتم مشاهدتها مسبقاًفي مجموعة البيانات:

```
start_time = time.time()

with torch.no_grad():

    best_network = Network()
    best_network.cuda()

best_network.load_state_dict(torch.load('/content/face_landmarks.pth')
)
    best_network.eval()
```

```
    images, landmarks = next(iter(valid_loader))

    images = images.cuda()
    landmarks = (landmarks + 0.5) * 224

    predictions = (best_network(images).cpu() + 0.5) * 224
    predictions = predictions.view(-1,68,2)

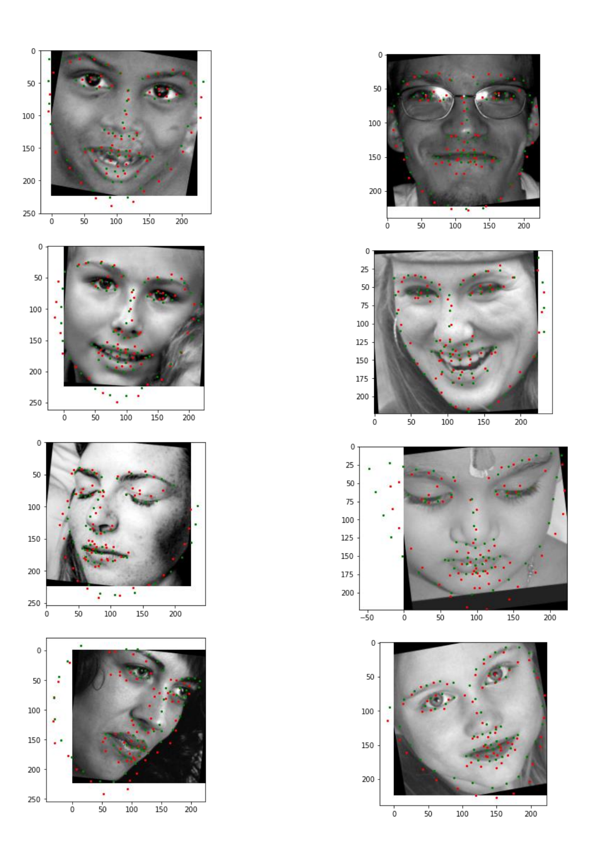
    plt.figure(figsize=(10,40))

    for img_num in range(8):
        plt.subplot(8,1,img_num+1)

plt.imshow(images[img_num].cpu().numpy().transpose(1,2,0).squeeze(),
cmap='gray')
        plt.scatter(predictions[img_num,:,0],
predictions[img_num,:,1], c = 'r', s = 5)
        plt.scatter(landmarks[img_num,:,0], landmarks[img_num,:,1], c
= 'g', s = 5)

print('Total number of test images: {}'.format(len(valid_dataset)))

end_time = time.time()
print("Elapsed Time : {}".format(end_time - start_time))
```



18) تصنيف الكلاب والقطط باستخدام Dog and Cat CNN Classification using CNN

مقدمة إلى CNN

تُستخدم الشبكات العصبية التلافيفية (CNN) بشكل أساسي لتصنيف الصور أو تحديد أوجه التشابهفي الأنماط بينها.

لذلك تستقبل الشبكة التلافيفية صورة ملونة عادية كمربع مستطيل يقاس عرضه وارتفاعه بعدد البكسل على طول تلك الأبعاد، وعمقها ثلاث طبقات، واحدة لكل حرففي RGB.

عندما تتحرك الصور عبر شبكة تلافيفية، يتم التعرف على أنماط مختلفة تمامًا مثل الشبكة العصبية العادية.

ولكن هنا بدلاً من التركيز على بكسل واحدفي كل مرة، تأخذ الشبكة التلافيفية رقع مربعة square patches من البكسلات وتمررها عبر فلتر filter.

هذا الفلتر هو أيضًا مصفوفة مربعة أصغر من الصورة نفسها، ويساوي حجمها الرقعة patch. ويسمى أيضًا الكيرنل kernel.

لنبدأ الآن باستيراد المكتبات

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import cv2

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten, Dropout, Activation,
Conv2D
```

نحتاج إلى تدريب نموذج أولاً حتى نتحقق من بيانات التدريبفي الكود أدناه، نقوم بالتكرار خلال جميع الصور الموجودةفي مجلد التدريب ثم نقوم بتقسيم اسم الصورة باستخدام المحدد "."

لدينا أسماء مثل dog.0، وdog.1، وcat.2، وما إلى ذلك.. ومن ثم بعد التقسيم، سنحصل على نتائج مثل "dog "، و "cat " كقيمة فئة للصورة. لجعل هذا المثال أكثر سهولة، سننظر إلى الكلب على أنه "1" والقط على أنه "0".

الآن كل صورة هي في الواقع مجموعة من البكسل، فكيف نجعل جهاز الكمبيوتر الخاص بنا يعرف ذلك. إنه بسيط يحول كل تلك البكسلات إلى مصفوفة.

لذلك سنستخدم هنا مكتبة cv2 لقراءة صورتنافي مصفوفة وأيضًا ستقرأ كصورة ذات مقياس رمادي.

```
train_dir = # your path to train dataset
path = os.path.join(main_dir,train_dir)

for p in os.listdir(path):
    category = p.split[0](".")
    img_array = cv2.imread(os.path.join(path,p),cv2.IMREAD_GRAYSCALE)
    new_img_array = cv2.resize(img_array, dsize=(80, 80))
    plt.imshow(new_img_array,cmap="gray")
    break
```

حسنًا، كان الرمز أعلاه أكثر لفهم الغرض. الآن سوف نصل إلى الجزء الحقيقي من البرمجة هنا.

أعلن عن مجموعة التدريب الخاصة بك X والمصفوفة المستهدفة y. هنا ستكون X عبارة عن مجموعة من وحدات البكسل وستكون y هي القيمة 0 أو 1 للإشارة إلى كلب أو قطة. اكتب دالة تحويل إلى مطابقة الفئة "dog " أو "cat" في 1 و0.

قم بإنشاء دالة create_test_data تأخذ كل صور التدريب في حلقة. يتحول إلى مجموعة صور. قم بتغيير حجم الصورة إلى 80 × 80. قم باضافة صورةفي مجموعة X. وقم باضافة قيمة الفئةفي المصفوفة y.

```
X[] =
y[] =
convert = lambda category: int(category == 'dog')
def create_test_data(path):
    for p in os.listdir(path):
        category = p.split[0](".")
        category = convert(category)
        img_array                                                          =
cv2.imread(os.path.join(path,p),cv2.IMREAD_GRAYSCALE)
        new_img_array = cv2.resize(img_array, dsize=(80, 80))
```

```
        X.append(new_img_array)
        y.append(category)
```

الآن استدعي الدالة، ولكن أيضًا لاحقًا حول X وy إلى مصفوفة numpy، وعلينا أيضًا إعادة تشكيل X بالكود أدناه:

```
create_test_data(path)
X = np.array(X).reshape(80,80,1 ,1-)
y = np.array(y)
```

إذا رأيت قيم X، يمكنك رؤية مجموعة متنوعة من القيم بين 0ـ255. ذلك لأن كل بكسل له كثافة مختلفة من الأبيض والأسود. ولكن مع وجود مجموعة كبيرة من القيم، يصبح من الصعب على نموذج التدريب التعلم (حفظه أحيانًا).

كيف تحل هذا وقد خمنت ذلك بشكل صحيح. يمكنك تسوية normalize البيانات. يمكننا استخدام تسوية Keras هنا أيضًا. لكننا نعلم جيداً أن جميع القيم لها نطاق يتراوح بين 0ـ255، لذا يمكننا فقط تقسيمها على 255 والحصول على جميع القيم بقياس ما بين 0ـ1

هذا ما فعلناه أدناه. يمكنك تخطي هذه الخطوة لمعرفة الفرق بين الدقة. لا تصدق كل ما أقوله. جرب وانظر بنفسك:

```
#Normalize data
X = X/255.0
```

تدريب النموذج

```
model = Sequential()
 #Adds a densely-connected layer with 64 units to the model:
model.add(Conv2D(64,(3,3),   activation   =   'relu',   input_shape   =
X.shape[1:]))
model.add(MaxPooling2D(pool_size = (2,2)))
 #Add another:
model.add(Conv2D(64,(3,3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2)))


model.add(Flatten())
model.add(Dense(64, activation='relu'))
 #Add a softmax layer with 10 output units:
```

```
model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer="adam,"
              loss='binary_crossentropy,'
              metrics=['accuracy']( 
```

الآن سوف نلائم نموذجنا مع بيانات التدريب.

ال**فترات Epochs**: كم مرة سوف يمر نموذجنا بالبيانات.

حجم الدُفعة Batch size: مقدار البيانات التي تريد تمريرها عبر النموذج دفعة واحدة.

Validation_split: ما مقدار البيانات (في هذه الحالة 20٪) الذي ستحتاجه للتحقق من خطأ التحقق من الصحة.

```
model.fit(X,    y,    epochs=10,    batch_size=32,    validation_split=0.2)
```

```
1 Train on 20000 samples, validate on 5000 samples
2 Epoch 1/10
3 20000/20000 [==============================] - 16s 790us/step - loss: 0.610
4 Epoch 2/10
5 20000/20000 [==============================] - 14s 679us/step - loss: 0.498
6 Epoch 3/10
7 20000/20000 [==============================] - 14s 679us/step - loss: 0.450
8 Epoch 4/10
9 20000/20000 [==============================] - 14s 680us/step - loss: 0.405
10 Epoch 5/10
11 20000/20000 [==============================] - 14s 679us/step - loss: 0.367
12 Epoch 6/10
13 20000/20000 [==============================] - 14s 679us/step - loss: 0.318
14 Epoch 7/10
15 20000/20000 [==============================] - 14s 680us/step - loss: 0.270
16 Epoch 8/10
17 20000/20000 [==============================] - 14s 681us/step - loss: 0.215
18 Epoch 9/10
19 20000/20000 [==============================] - 14s 679us/step - loss: 0.164
20 Epoch 10/10
21 20000/20000 [==============================] - 14s 680us/step - loss: 0.122
```

حان الوقت الآن للتنبؤ PREDICT أخيرًا، لذا قم بتغذية نموذج CNN الخاص بك ببيانات الاختبار للتنبؤ.

```
predictions = model.predict(X_test)
```

نحن نقرب النتيجة هنا حيث استخدمنا دالة sigmoid وحصلنا على قيم الاحتمالفي مجموعة البيانات المتوقعة لدينا:

```
predicted_val = [int(round(p[0])) for p in predictions]
```

الآن عليك أن تجعل إطار بيانات الإرسال لإرسال مجموعة النتائج الخاصة بك.

```
submission_df = pd.DataFrame({'id':id_line, 'label':predicted_val})
```

اكتب إطار البيانات الخاص بك إلى ملف csv:

```
submission_df.to_csv("submission.csv", index=False)
```

19) تحليل المشاعر على تويتر Twitter Sentiment Analysis

تحليل المشاعر على Twitter هو عملية تحديد وتصنيف التغريدات بشكل حسابي في جزء من النص، لا سيما من أجل تحديد ما إذا كان موقف الكاتب تجاه موضوع معين، أو منتج معين، وما إلى ذلك، إيجابيًا أم سلبيًا أم محايداً.

في هذه المقالة سأقوم بتحليل المشاعر على تويتر باستخدام Natural Language Processing باستخدام مكتبة nltk مع بايثون.

تحليل المشاعر على تويتر

لنبدأ باستيراد المكتبات:

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.model_selection import train_test_split # function for splitting data to train and test sets

import nltk

from nltk.corpus import stopwords

from nltk.classify import SklearnClassifier


from wordcloud import WordCloud,STOPWORDS

import matplotlib.pyplot as plt
```

تنزيل مجموعة البيانات

```
data = pd.read_csv('Sentiment.csv')

 #Keeping only the neccessary columns

data = data[['text','sentiment']]
```

بادئ ذي بدء، تقسيم مجموعة البيانات إلى مجموعة تدريب واختبار. مجموعة الاختبار هي 10٪ من مجموعة البيانات الأصلية.

بالنسبة لهذا التحليل الخاص، أسقطت التغريدات المحايدة، حيث كان هدفي هو التمييز بين التغريدات الإيجابية والسلبية فقط.

```
 #Splitting the dataset into train and test set

train, test = train_test_split(data,test_size = 0.1)

 #Removing neutral sentiments

train=train[train.sentiment!= "Neutral"]
```

كخطوة تالية، قمت بفصل التغريدات الإيجابية والسلبية لمجموعة التدريب من أجل رسم الكلمات المضمنة بسهولة.

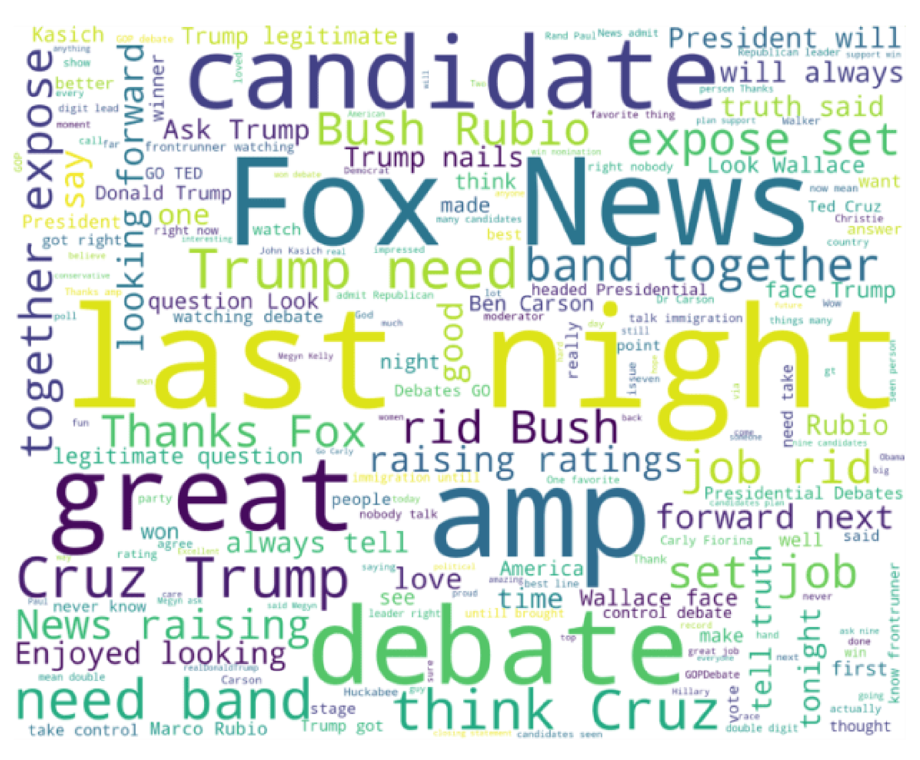
بعد ذلك قمت بتنظيف النص من علامات التصنيف والإشارات والروابط. أصبحوا الآن جاهزين لرسم WordCloud الذي يعرض فقط الكلمات الأكثر تأكيداً للتغريدات الإيجابية والسلبية.

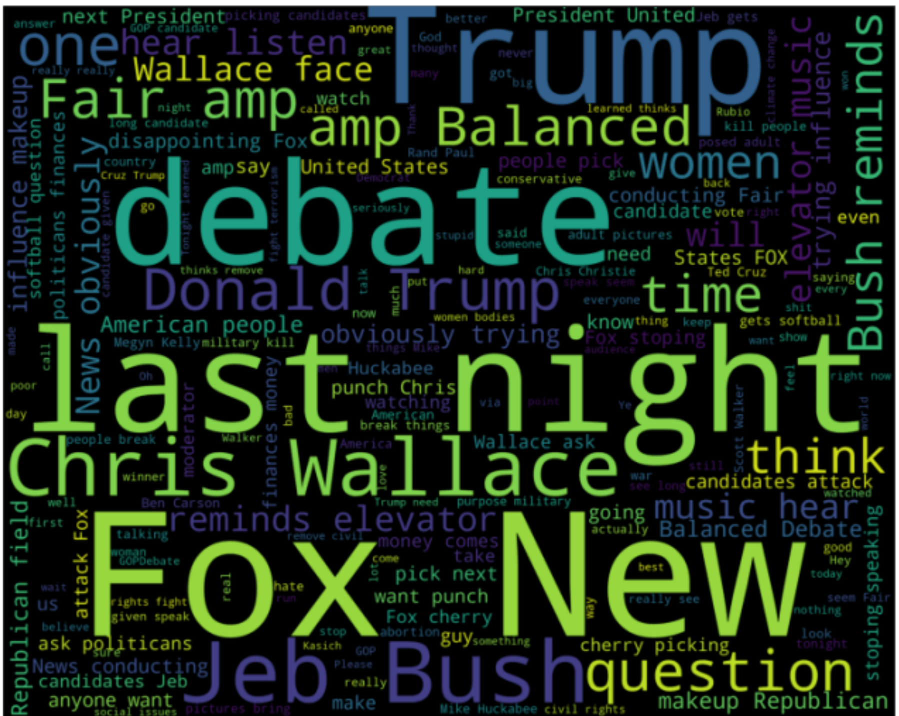
```
train_pos = train[ train['sentiment'] == 'Positive']
train_pos = train_pos['text']
train_neg = train[ train['sentiment'] == 'Negative']
train_neg = train_neg['text']


def wordcloud_draw(data, color = 'black'):
    words = ' '.join(data)
    cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and not word.startswith('#')
                                and word != 'RT'
                                ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
                      height=2000
                      ).generate(cleaned_word)
    plt.figure(1,figsize=(13, 13))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()


print("Positive words")
wordcloud_draw(train_pos,'white')
print("Negative words")
```

```
wordcloud_draw(train_neg)
```





من المثير للاهتمام ملاحظة الكلمات والعبارات التاليةفي مجموعة الكلمات الإيجابية: truth، strong، legitimate، together، love، job

في تفسيري، يميل الناس إلى الاعتقاد بأن مرشحهم المثالي صادق truthful وشرعي legitimate وفوق الخير والشر.

في الوقت نفسه، تحتوي التغريدات السلبية على كلمات مثل: influence، news، elevatormusic، disappointing، softball، makeup، cherry picking، trying

حسب فهمي، فات الناس التمثيل الحاسم واعتبروا المرشحين الموبخين ضعيفين للغاية ويقطفون الكرز cherry picking.

بعد التحويل إلى رسم البيانات vizualization، قمت بإزالة علامات التوقف stopwords والإشارات والروابط وكلمات الإيقاف من مجموعة التدريب.

Stop Words : هي الكلمات التي لا تحتوي على أهمية مهمة لاستخدامهافي استعلامات البحث.

عادةً ما يتم تصفية هذه الكلمات من استعلامات البحث لأنها تُرجع قدرًا هائلاً من المعلومات غير الضرورية. (the، for، this، إلخ.)

```
tweets =[]
stopwords_set = set(stopwords.words("english"))


for index, row in train.iterrows:()
    words_filtered = [e.lower() for e in row.text.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in stopwords_set]
    tweets.append((words_without_stopwords, row.sentiment))


test_pos = test[ test['sentiment'] == 'Positive']
```

```
test_pos = test_pos['text']
test_neg = test[ test['sentiment'] == 'Negative']
test_neg = test_neg['text']
```

كخطوة تالية، قمت باستخراج الميزات المسماة باستخدام nltk lib، أولاً عن طريق قياس التوزيع المتكرر واختيار المفاتيح الناتجة.

```
#Extracting word features
def get_words_in_tweets(tweets):
    all[] =
    for (words, sentiment) in tweets:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_tweets(tweets))

def extract_features(document):
    document_words = set(document)
    features{} =
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

بموجب هذا قمت برسم الكلمات الأكثر انتشارًا. تتركز معظم الكلمات حول ليالي المناظرة debate nights.

```
wordcloud_draw(w_features)
```

باستخدام Classifier NaiveBayes nltk، قمت بتصنيف ميزات كلمات التغريدة المستخرجة.

```
#Training the Naive Bayes classifier
```

```
training_set = nltk.classify.apply_features(extract_features,tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

أخيرًا، باستخدام مقاييس غير ذكية، حاولت قياس كيفية تسجيل خوارزمية المصنف.

```
neg_cnt = 0
pos_cnt = 0
for obj in test_neg :
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Negative') :
        neg_cnt = neg_cnt + 1
for obj in test_pos :
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Positive') :
        pos_cnt = pos_cnt + 1

print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))
```

```
[Negative]: 842/795
[Positive]: 220/74
```

20) نموذج التنبؤ بالكلمة التالية Next Word Prediction Model

تقدم معظم لوحات المفاتيح في الهواتف الذكية ميزات التنبؤ بالكلمة التالية؛ يستخدم google أيضًا توقع الكلمة التالية استنادًا إلى سجل التصفح الخاص بنا. لذلك يتم أيضًا تخزين البيانات المحملة مسبقًا في وظيفة لوحة المفاتيح بهواتفنا الذكية للتنبؤ بالكلمة التالية بشكل صحيح.في هذه المقالة، سأقوم بتدريب نموذج التعلم العميق للتنبؤ بالكلمة التالية باستخدام بايثون. سأستخدم مكتبة Tensorflow وKeras في بايثون لنموذج التنبؤ بالكلمة التالية.

لصنع نموذج توقع الكلمة التالية، سأقوم بتدريب شبكة عصبية متكررة (RNN). فلنبدأ بهذه المهمة الآن دون إضاعة أي وقت.

نموذج التنبؤ بالكلمة التالية

للبدء بنموذج التنبؤ بالكلمة التالية، دعنا نستورد بعض المكتبات التي نحتاجها لهذه المهمة:

```
import numpy as np
from nltk.tokenize import RegexpTokenizer
from keras.models import Sequential, load_model
from keras.layers import LSTM
from keras.layers.core import Dense, Activation
from keras.optimizers import RMSprop
import matplotlib.pyplot as plt
import pickle
import heapq
```

كما قلت سابقًا، تستخدم Google سجل التصفح الخاص بنا لعمل تنبؤات بالكلمة التالية، ويتم تدريب الهواتف الذكية وجميع لوحات المفاتيح المدربة على التنبؤ بالكلمة التالية باستخدام بعض البيانات. لذلك سأستخدم أيضًا مجموعة بيانات. يمكنك تنزيل مجموعة البيانات من هنا.

الآن دعنا نحمّل البيانات ونلقي نظرة سريعة على ما سنعمل معه:

```
path = '1661-0.txt'
text = open(path).read().lower()
print('corpus length:', len(text))
```

```
corpus length: 581887
```

سأقوم الآن بتقسيم مجموعة البيانات إلى كل كلمة بالترتيب ولكن دون وجود بعض الأحرف الخاصة.

```
tokenizer = RegexpTokenizer(r'w+')
```

```
words = tokenizer.tokenize(text)
```

```
['project', 'gutenberg', 's', 'the', 'adventures', 'of', 'sherlock', 'holmes', 'by', ............................ , 'our',
'email', 'newsletter', 'to', 'hear', 'about', 'new', 'ebooks']
```

الآن ستكون العملية التالية هي تنفيذ هندسة الميزات feature engineering في بياناتنا. لهذا الغرض، سنطلب قاموسًا يحتوي على كل كلمة في البيانات ضمن قائمة الكلمات الفريدة كمفتاح key، وهي أجزاء مهمة كقيمة value.

```
unique_words = np.unique(words)

unique_word_index = dict((c, i) for i, c in enumerate(unique_words))
```

هندسة الميزات

تعني هندسة الميزات أخذ أي معلومات لدينا حول مشكلتنا وتحويلها إلى أرقام يمكننا استخدامها لبناء مصفوفة الميزات الخاصة بنا. إذا كنت تريد برنامجًا تعليميًا مفصلاً عن هندسة الميزات، فيمكنك تعلمه من هنا.

سأحدد هنا طول الكلمة الذي سيمثل عدد الكلمات السابقة التي ستحدد كلمتنا التالية. سأحدد الكلمات السابقة للحفاظ على الكلمات الخمس السابقة والكلمات التالية المقابلة لها في قائمة الكلمات التالية.

```
WORD_LENGTH = 5

prev_words[] =

next_words[] =

for i in range(len(words) - WORD_LENGTH):

    prev_words.append(words[i:i + WORD_LENGTH])

    next_words.append(words[i + WORD_LENGTH])

print(prev_words[0])

print(next_words[0])
```

```
['project', 'gutenberg', 's', 'the', 'adventures']
```

الآن سوف أقوم بإنشاء مصفوفتين فارغتين x لتخزين الميزات وy لتخزين التسمية المقابلة لها. سأكرر x وy إذا كانت الكلمة متاحة بحيث يصبح الموضع المقابل 1.

```
X    =    np.zeros((len(prev_words),    WORD_LENGTH,    len(unique_words)),
dtype=bool)

Y = np.zeros((len(next_words), len(unique_words)), dtype=bool)

for i, each_words in enumerate(prev_words):

    for j, each_word in enumerate(each_words):

        X[i, j, unique_word_index[each_word]] = 1
```

```
    Y[i, unique_word_index[next_words[i]]] = 1
```

الآن قبل المضي قدمًا، ألق نظرة على سلسلة واحدة من الكلمات:

```
print(X[0][0])
```

```
[False False False ... False False False]
```

بناء الشبكة العصبية المتكررة

كما ذكرت سابقًا، سأستخدم الشبكات العصبية المتكررة لنموذج التنبؤ بالكلمة التالية. هنا سأستخدم نموذج LSTM، وهو RNN قوي جداً.

```
model = Sequential()

model.add(LSTM(128, input_shape=(WORD_LENGTH, len(unique_words))))

model.add(Dense(len(unique_words)))

model.add(Activation('softmax'))
```

تدريب نموذج توقع الكلمة التالية

سأقوم بتدريب نموذج التنبؤ بالكلمة التاليةفي 20 حقبة:

```
optimizer = RMSprop(lr=0.01)

model.compile(loss='categorical_crossentropy',     optimizer=optimizer,
metrics=['accuracy'])

history   =   model.fit(X,   Y,   validation_split=0.05,   batch_size=128,
epochs=2, shuffle=True).history
```

لقد قمنا الآن بتدريب نموذجنا بنجاح، قبل المضي قدمًافي تقييم نموذجنا، سيكون من الأفضل حفظ هذا النموذج لاستخدامنافي المستقبل.

```
model.save('keras_next_word_model.h5')

pickle.dump(history, open("history.p", "wb"))

model = load_model('keras_next_word_model.h5')

history = pickle.load(open("history.p", "rb"))
```

تقييم نموذج التنبؤ بالكلمة التالية

دعنا الآن نلقي نظرة سريعة على كيفية تصرف نموذجنا بناءً على تغييرات دقته وخسارته أثناء التدريب:

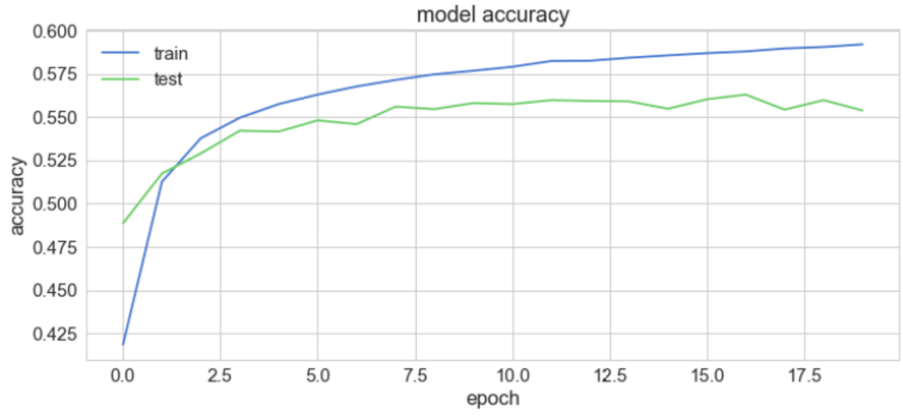
```
plt.plot(history['acc'])

plt.plot(history['val_acc'])

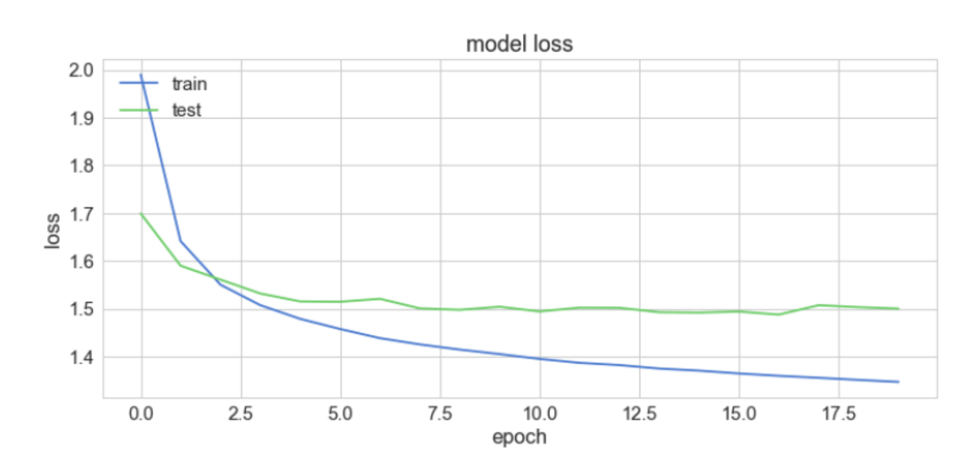
plt.title('model accuracy')

plt.ylabel('accuracy')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```




```
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```



اختبار نموذج التنبؤ بالكلمة التالية

فلنقم الآن ببناء برنامج بايثون للتنبؤ بالكلمة التالية باستخدام نموذجنا المُدرَّب. لهذا، سأحدد بعض الدوال الأساسية التي سيتم استخدامهافي العملية.

```
def prepare_input(text):
    x = np.zeros((1, SEQUENCE_LENGTH, len(chars)))
```

```
    for t, char in enumerate(text):
        x[0, t, char_indices[char]] = 1.

    return x
```

الآن قبل المضي قدمًا، دعنا نختبر الدالة، تأكد من استخدام دالة lower() أثناء إعطاء الإدخال:

```
prepare_input("This is an example of input for our LSTM".lower())
```

```
array([[[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
...,
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.]]])
```

لاحظ أن التسلسلات يجب أن تتكون من 40 حرفًا (وليس كلمات) حتى نتمكن من وضعها بسهولةفي موتر من الشكل (57، 40، 1). قبل المضي قدمًا، فلنتحقق مما إذاكانت الدالة التي تم إنشاؤها تعمل بشكل صحيح.

```
def prepare_input(text):
    x = np.zeros((1, WORD_LENGTH, len(unique_words)))
    for t, word in enumerate(text.split()):
        print(word)
        x[0, t, unique_word_index[word]] = 1
    return x
prepare_input("It is not a lack".lower())
```

```
array([[[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
...,
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.],
[ 0., 0., 0., ..., 0., 0., 0.]]])
```

الآن سأقوم بإنشاء دالة لإرجاع العينات:

```
def sample(preds, top_n=3):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds)
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
```

```
    return heapq.nlargest(top_n, range(len(preds)), preds.take)
```

والآن سأقوم بإنشاء دالة للتنبؤ بالكلمة التالية:

```
def predict_completion(text):
    original_text = text
    generated = text
    completion'' =
    while True:
        x = prepare_input(text)
        preds = model.predict(x, verbose=0)[0]
        next_index = sample(preds, top_n=1)[0]
        next_char = indices_char[next_index]
        text = text[1:] + next_char
        completion += next_char

        if len(original_text + completion) + 2 > len(original_text) and
next_char:' '==
            return completion
```

تم إنشاء هذه الدالة للتنبؤ بالكلمة التالية حتى يتم إنشاء مساحة. سيفعل ذلك عن طريق تكرار الإدخال، والذي سيطلب نموذج RNN الخاص بنا ويستخرج مثيلات منه. الآن سأقوم بتعديل الدالة المذكورة أعلاه للتنبؤ بأحرف متعددة:

```
def predict_completions(text, n=3):
    x = prepare_input(text)
    preds = model.predict(x, verbose=0)[0]
    next_indices = sample(preds, n)
    return [indices_char[idx] + predict_completion(text[1:] +
indices_char[idx]) for idx in next_indices]
```

الآن سأستخدم التسلسل المكون من 40 حرفًا والذي يمكننا استخدامه كأساس لتوقعاتنا.

```
quotes =[
"   It is not a lack of love, but a lack of friendship that makes
unhappy marriages,".
"   That which does not kill us makes us stronger,".
```

```
"    I'm not upset that you lied to me, I'm upset that from now on I can't believe you,".

"    And those who were seen dancing were thought to be insane by those who could not hear the music,".

"    It is hard enough to remember my opinions, without also remembering my reasons for them"!

[
```

الآن أخيرًا، يمكننا استخدام النموذج للتنبؤ بالكلمة التالية:

```
for q in quotes:

    seq = q[:40].lower()

    print(seq)

    print(predict_completions(seq, 5))

    print()
```

```
it is not a lack of love, but a lack of
['the ', 'an ', 'such ', 'man ', 'present, ']
that which does not kill us makes us str
['ength ', 'uggle ', 'ong ', 'ange ', 'ive ']
i'm not upset that you lied to me, i'm u
['nder ', 'pon ', 'ses ', 't ', 'uder ']
and those who were seen dancing were tho
['se ', 're ', 'ugh ', ' servated ', 't ']it is hard enough to remember my opinion
[' of ', 's ', ', ', 'nof ', 'ed ']
```

أتمنى أن تكون قد أحببت هذا المقال من نموذج التنبؤ بالكلمة التالية.

20

Deep Learning Projects with Python

By

Aman Kharwal

Translated Into Arabic by

Dr. Alaa Taima